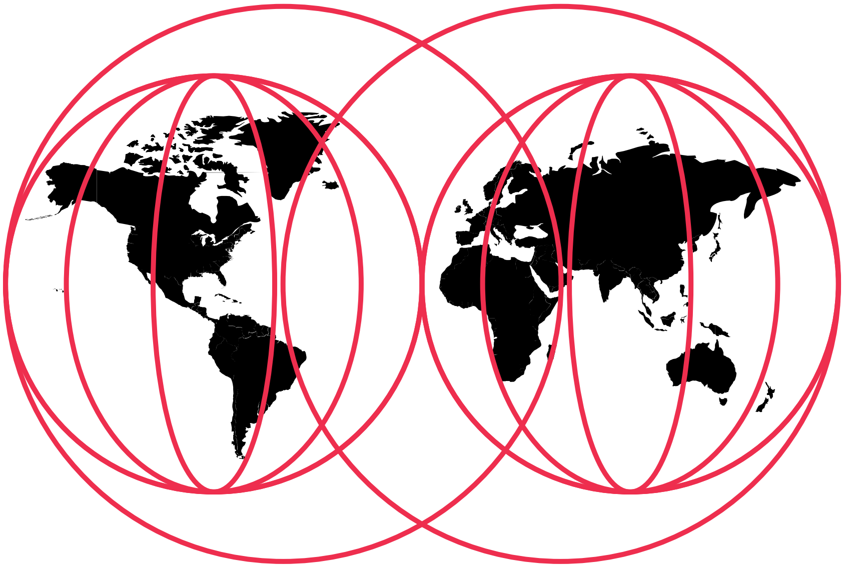
Lotus Sametime Application Development Guide

David Morrison, John Fiola, Ollie B. Rashid, Steve Daly



International Technical Support Organization

www.redbooks.ibm.com

SG24-5651-00

SG24-5651-00

IBM International Technical Support Organization

Lotus Sametime Application Development Guide

May 2000

First Edition (May 2000)

This edition applies to Lotus Domino Release 5.0.2 and Lotus Sametime 1.5.

Comments may be addressed to: IBM Corporation, International Technical Support Organization
1 Charles Park, 5602D
Cambridge
MA 02142

Contents

Preface

This redbook explains how to incorporate Lotus Sametime into your Notes/Domino and Web environments. We discuss in detail how to Sametime enable Lotus Notes applications using the LotusScript and Java toolkits and how to embed Sametime technology into your Web sites. In the initial chapters we describe the functionality of Sametime and discuss some of the concepts associated with instant messaging. We then take you through the Sametime API and start incorporating Who is Online awareness into a Domino application, finally adding in Who is Here awareness to your applications. We then discuss how to incorporate the Sametime meeting services into your applications before describing a detailed case study where Sametime can be used.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Center at Lotus in Cambridge, Massachusetts, USA.

David Morrison is an International Technical Support Specialist for Notes and Domino at the International Technical Support Organization Center at Lotus Development, Cambridge, Massachusetts. He manages projects whose objective it is to produce redbooks on all areas of Domino. Before joining the ITSO in 1999, he was a senior Lotus Notes consultant working for IBM e-business services in the United Kingdom.

John Fiola is a Senior Programmer Analyst in the Advanced Technologies Group in Lotus Development's Information Services (IS) section. He advises IS development groups on using and deploying new Lotus technologies and has been working with Sametime in this capacity since the product's conception. Prior to that, he was an application developer in Lotus's Software Problem Reporting system.

Ollie B. Rashid is an Instructor/Curriculum Developer with Databeam, an IBM subsidiary. She began to work with Sametime shortly after its 1.0 release, creating application development and end-user training materials for the product.

Steve Daly is a Manager with the Cincinnati, OH branch of marchFIRST, Inc. As a member of the Collaborative Technologies practice, Steve is

responsible for helping companies deploy groupware technologies based on Lotus Notes and Domino. He worked with Lotus Development Corporation on the development of Sametime 1.0 and LearningSpace Anytime 3.0.

A number of people have provided support and guidance. In particular, we would like to thank the following people. Unless otherwise noted, they are from Lotus Cambridge:

- Mike Dempsey - input on Sametime security
- John Conrad - input on API architecture
- Benjamin Gold - general Sametime knowledge
- Tony Payne - general Sametime knowledge
- Malissa Sullivan - general Sametime knowledge
- Bas van der Hoek - how we code around the LSX error
- Haim Schneider - explanation of how the Connect Client dlls work with the LSX
- Bob Anderson - Sametime-enabled R5 discussion
- Dave Wetherell - Java and JavaScript coding
- Ron Pontrich (Whittman-Hart) - information on Sametime development
- Richard Blake and Rob Fox - participation in screen captures

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found at the back of this book to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com/`

Send your comments in an Internet note to `redbook@us.ibm.com`

Chapter 1
What is Lotus Sametime

As today's workforce becomes more dispersed and virtual teams become the norm, businesses are looking for ways to quickly connect their workers for collaborative efforts. Lotus Sametime is the product that delivers a network-based, real-time communication and collaboration solution to today's global business. A pivotal component to achieve the Lotus ideal of collaboration, Sametime establishes a virtual office environment where workers can easily locate and communicate with colleagues, customers, suppliers, and others.

The Sametime product family consists of these components:

- Sametime Server
- Client Package
- Online Meeting Center
- Sametime-enabled Lotus Notes databases
- Application Development Toolkits
- Server Administration tool

In this chapter we discuss these components, explain Sametime's capabilities, and describe the product's power to transform any work environment.

The capabilities of Sametime

Sametime meets the need for workers to connect and communicate instantly on issues that are critical to business success. Sametime fulfills this need by establishing an online community and providing these key user capabilities:

- Awareness
- Conversation
- Object sharing

Awareness

Awareness allows users to quickly locate others who are online in a Sametime community. It can be compared to peripheral vision in that users are made conscious or others' presence without removing themselves from their current work environment or process. To monitor the online presence of others, users specify to Sametime the names of those they want to track. In real-time, Sametime detects and updates the online status of users that it is monitoring.

Conversation

The Sametime *conversation* capability allows users to communicate with each other using text chat. When you see that someone is online (Awareness), you can use your keyboard to send them an instant message and engage in a one-on-one chat. A conference chat can be created by inviting a third party into an existing one-on-one chat or by initiating a chat with a group of users.

Object sharing

Sametime includes an online meeting facility with which users can schedule and attend online meetings. Sametime also provides a venue for users to instantly launch impromptu meetings. The interaction and collaboration in online meetings is much richer when meeting participants have shared access to an object (document, application, and so on). For that reason, Sametime provides two *object sharing* tools: application sharing and an electronic whiteboard.

Available for scheduled and instant meetings, the *application sharing* tool allows users to share any 32-bit Windows application. This takes place without everyone having the application installed locally. The "host", or sharer, of the application can let others control the application during the meeting. This tool is great for team collaboration efforts such as presentation creation, document reviews and project status updates.

Application sharing can be especially useful for quickly gathering information from several parties. For example, a sales manager can start an instant meeting with three salespeople to collect monthly sales figures. In the meeting, the manager shares a spreadsheet and switches control to each salesperson, allowing them to enter their sales data directly into the spreadsheet. At the end of the meeting, the manager saves the spreadsheet and uses it as a report.

The electronic *whiteboard* can be used to present files such as full-length presentations. Or, you can share a blank whiteboard during meetings so that participants can mark on it just as they would on a chalk board. The whiteboard contains annotation tools that can be used to enter text and draw objects.

The Sametime server

The Sametime server is the *source* of Sametime's user capabilities. This server can be installed as a standalone product in a Web-only environment, or it can be integrated into a Domino environment. When installed as a standalone solution, Sametime provides its capabilities in a synchronous real-time mode. However, when optionally installed as a complement to Domino, Sametime provides a powerful synchronous and asynchronous solution.

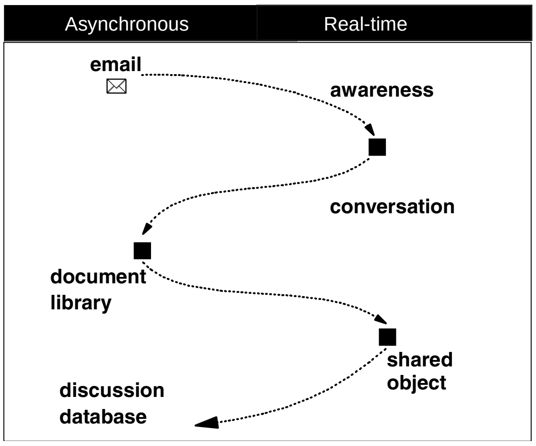
The power of Sametime

The unique power of Sametime is its ability to move between asynchronous and real-time communications while adding value to the user's experience. Let's look at a business scenario where Sametime demonstrates this power.

Suppose you receive an urgent e-mail from a customer with a serious software support issue. You must provide an immediate and accurate solution to this issue or lose the customer account. Using e-mail, you could exchange endless messages with an internal expert seeking answers to the issue. The downtime between messages could grow into hours, leaving the customer even more dissatisfied.

Using Sametime installed in a Domino environment, you could get much faster results than with e-mail. First, with awareness you can quickly locate an internal expert knowledgeable about the customer issue and determine if that person is available for discussion. Seeing that he is available, you send an instant message asking him a few direct questions. During the chat session, you decide to invite another internal expert to help with the issue. This conference chat leads to a discussion about a document stored in the Customer Service database. Using Sametime's instant meeting feature, you access that Notes database, open the document and share it with the experts online. In real-time, the group reviews the document. Based on the online review, necessary document changes are made. You save the document and repost it to the Customer Service database. In minutes you have created a document that accurately addresses the customer's specific needs. You can now forward that document to the customer or discuss it in a phone conversation.

While resolving the customer issue, Sametime allowed you to seamlessly move between the asynchronous environments of e-mail and Notes databases to real-time chat and document sharing. That is the power of Sametime.



The Sametime product family

The Sametime server

The Sametime server supports the services that deliver awareness, conversation and object sharing to users in the Sametime community. A Sametime server can be installed as a standalone in a Web-only environment or it can be integrated into a Domino domain. You can deploy Sametime servers in a single or multiple-server configuration.

Clients supported by the Sametime server include Web browsers (Internet Explorer v4.x and later and Netscape v4.06 and later), Lotus Notes v4.6.2 and later and any T.120 client, including Microsoft Net Meeting. T.120 clients are those that comply to the T.120 standard established by the International Telecommunications Union (ITU). This family of open standards contains a series of communication and application protocols and services that provide support for real-time, multipoint communication.

When installed in a Web-only environment, the Sametime server can be accessed only by Web browser clients. Lotus Notes and Web browsers can access Sametime servers installed in a Domino environment. The Sametime server supports these services:

- Community Services
- Meeting Services
- Web Application Services
- Domino Application Services

Community Services

Community Services handles client login requests and user name resolution on the Sametime server. The Sametime Connect client application, a part of the Sametime Client Package, is supported by Community Services. This client application provides the online awareness and conversation capabilities.

Client connections to the server and Sametime Address Book browsing are also handled by Community Services. If Sametime is installed in a Domino domain, browsing of the Notes Address Book is supported. "Who is Online" and "Who is Here" awareness in Sametime Connect and Sametime-enabled applications and databases are supported by Community Services.

"Who is Online" awareness, also called People-based awareness, provides users with a list of all users who are logged into the Sametime community. *"Who is Here*" awareness, also referred to as Place-based awareness, supplies a list of online users in a specific place. A place can be a database, a document, an e-mail message, and so on.

Meeting Services

Meeting Services employs T.120 data-conferencing technology to support online meetings. This server component provides real-time collaboration using the application sharing and electronic whiteboard tools. During meetings, these tools, which are Java applets, are served to clients by Meeting Services.

Meeting Services also provides the following support for online meetings:

- Maintains multiple client connections
- Disseminates meeting data to clients
- Maintains a list of active and scheduled online meetings
- Starts and stops meetings at the appropriate times

Web Application Services

Web Application Services contains an HTTP Web server and an engine that converts Lotus Notes constructs into HTML. The HTTP Web server in this component permits browser access to Sametime. Using Netscape Navigator (v4.06 or later) or Internet Explorer (v4.x or later), workers can connect to the Sametime server over the Internet or via a company's intranet.

Web Application Services support the directory and security services used by the Sametime server. When the Sametime server is installed in a Domino environment, replication services are also provided.

Domino Application Services

This server component enables Sametime to function with Domino servers in a Domino environment. The directory, security, and replication features of Sametime are supported by Domino Application Services. Sametime servers installed in a Domino environment include Sametime discussion, sample e-mail and sample document library applications that contain online awareness and chat capabilities. These applications are supported by Domino Application Services.

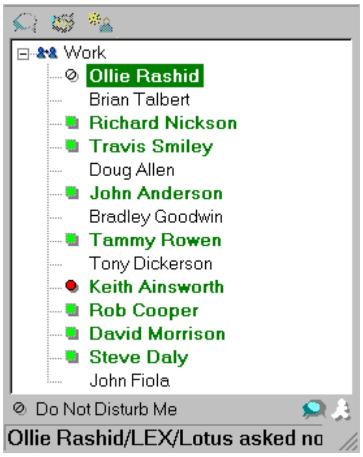
Sametime Client Package

The Sametime Client Package contains:

- Connect client
- Share an Application component

Connect Client

Written in C++, the *Connect* client is a standalone application that provides the online awareness and chat capabilities of the Sametime server. To use Connect, users must download the client software from the Sametime server and install it on their local computers. Once installed and configured, a Connect window similar to the one below appears on the user's desktop.



Ollie Rashid/LEX/Lotus asked no

Using the Connect menu commands, you can:

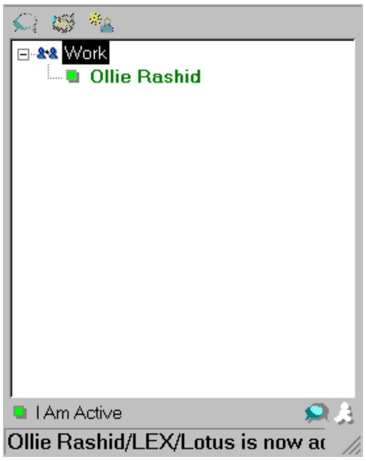
- Establish a desired list of users ("buddy list") to communicate with
- Monitor the online presence of users
- Initiate one-on-one and conference chats with online users
- Start an instant meeting using the Share an Application component

Establish a "Buddy List"

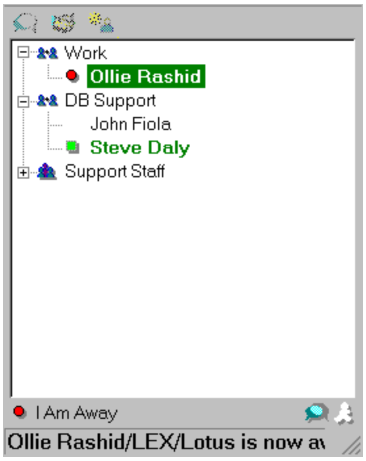
To monitor the online presence of other users, you must first add their names to Connect; this is known as establishing a *buddy list*. Do this by organizing the users into groups and placing them in the Connect window. These groups can be personal or public groups from the address book.

Personal groups are created by you and no one else in the organization can see or share them. *Public groups* are network-wide groups, usually departments, that contain the names of everyone assigned to that group. Public groups are usually created by the system administrator and cannot be manipulated. If you add a public group to your Connect window, all members of that group are added. Everyone in the organization can see and share public groups.

When you first install and open Connect, you see a default group named Work with your name in it. This group is your starting point for organizing your buddy list. Use the Add command in the People menu to add other users by typing their names or by choosing them from the available address books.



The Connect window below has three groups: Work, DB Support and Support Staff. Several names have been added to the personal group DB Support. Support Staff is a public group that contains many members. Because the group is collapsed, the members' names are not displayed.



Monitor the Online Presence of Users

Sametime provides online status icons which help users to quickly see who is online and available for collaboration. These online status icons appear beside users' names.

The status icons and their meanings are as follows:

- *I am Available*: When a user is online and available, a green square appears to the left of their name. The name appears in bold green text. If a user is online and available, you can engage them in chat or invite them to an online meeting.

  Rob Sinclair

- *I am Away:* When a user is online, but away from their computer, a red circle appears to the left of their name. The name appears in bold green text. If a user is away from their computer, you may send them a message or request them to join a meeting. The message or the request will appear on their desktop until they return to act on it.

  Dave Caldwell

- *Do Not Disturb*: When a user is online, but does not want to be disturbed, a circle with a slash appears to the left of their name. The name appears in bold green text. If you request interaction with a user having this status, you will receive a message telling you that the user has requested not to be disturbed. The user will not be aware of you effort to contact them.

  Brian Goodwin

- *Offline*: When a user is offline, a blank space appears to the left of their name. The name appears in thin black text. You cannot communicate with offline users.

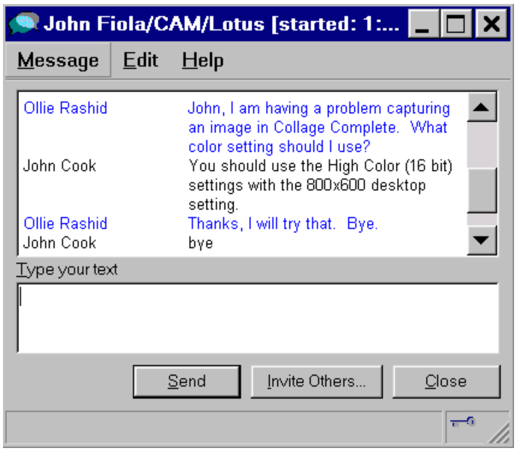
  Doug Mobley

Initiate One-on-One and Conference chats
Users can initiate one-on-one chats (instant messaging) by clicking on the name of an online user. To start the chat session, a user types a text message in the window and presses the Send button to forward the message to the chat partner. The partner responds with a message and the session continues. At any time during the chat session, either party can press the Invite Others button to bring others into the chat. When others join an existing one-on-one chat, a conference chat is created.

Users can click on the Close button at any time to disconnect from a chat session.

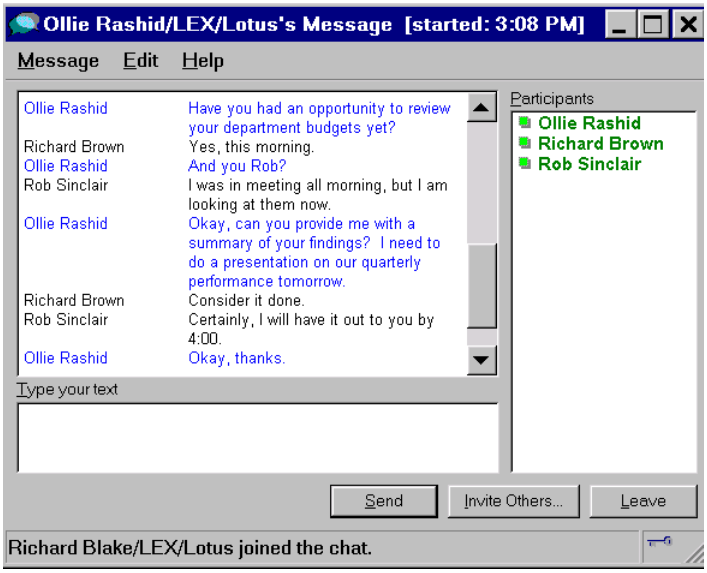
Note Connect interoperates with America Online's Instant Messenger (AIM). Like Connect, this product allows users to see who is currently online and to send quick private messages. Connect users can add AIM users to their Connect window and chat with them just as with other Connect users.

The following figure is an example of a chat window.



Ollie Rashid	John, I am having a problem capturing an image in Collage Complete. What color setting should I use?
John Cook	You should use the High Color (16 bit) settings with the 800x600 desktop setting.
Ollie Rashid	Thanks, I will try that. Bye.
John Cook	bye

You can also create a conference chat by *initially* selecting two or more people to chat with. This action brings up a conference chat window like the one in the following figure. The left pane of this window is where the chat actually takes place. The right pane displays a list of the chat participants and their online status.



Ollie Rashid	Have you had an opportunity to review your department budgets yet?
Richard Brown	Yes, this morning.
Ollie Rashid	And you Rob?
Rob Sinclair	I was in meeting all morning, but I am looking at them now.
Ollie Rashid	Okay, can you provide me with a summary of your findings? I need to do a presentation on our quarterly performance tomorrow.
Richard Brown	Consider it done.
Rob Sinclair	Certainly, I will have it out to you by 4:00.
Ollie Rashid	Okay, thanks.

Richard Blake/LEX/Lotus joined the chat.

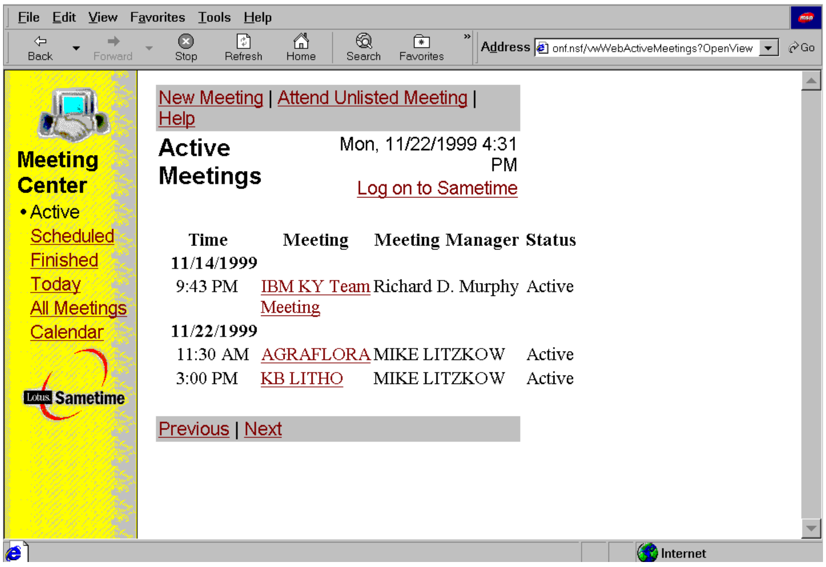
The Share an Application component

The Share an Application component of the Client Package allows users to initiate *instant meetings* and share 32-bit Windows applications with attendees. Sametime users can start instant meetings from the Connect window or from within an existing chat. Instant meetings can foster faster decision making and enable users to quickly engage in real-time collaboration on almost any project.

Online Meeting Center

The Sametime Online Meeting Center is a virtual environment where users schedule, find and attend real-time meetings. Like meeting down the hall in a conference room, you now can meet online with anyone, anywhere. In online meetings, attendees can share applications or present and modify files using the electronic whiteboard provided.

The Online Meeting Center can be accessed with a Web browser or Notes client. Several views are available to help users locate meetings. When the Online Meeting Center is opened initially, the Active Meetings view is shown.



Active Meetings

Mon, 11/22/1999 4:31 PM

Log on to Sametime

Time	Meeting	Meeting Manager	Status
11/14/1999			
9:43 PM	IBM KY Team Meeting	Richard D. Murphy	Active
11/22/1999			
11:30 AM	AGRAFLORA	MIKE LITZKOW	Active
3:00 PM	KB LITHO	MIKE LITZKOW	Active

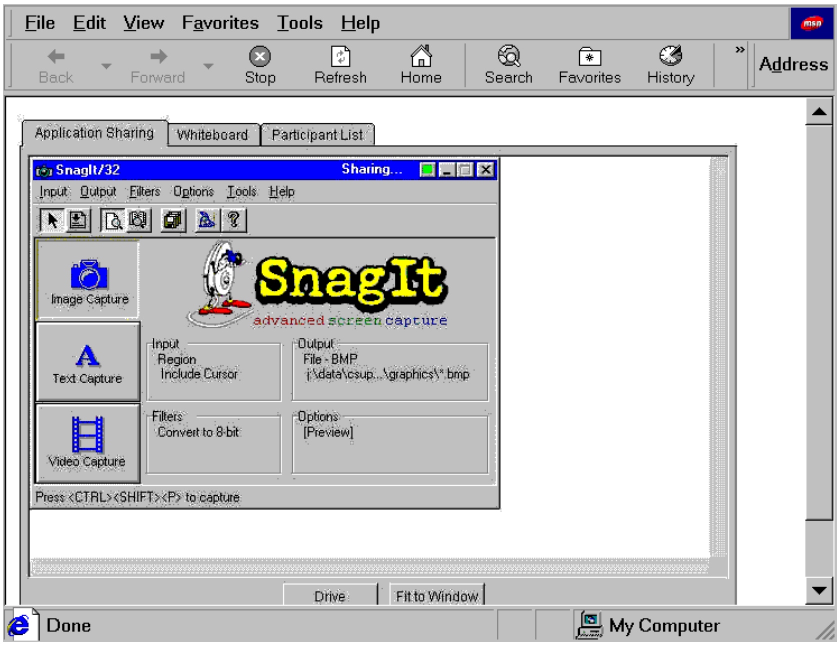
The Meeting Center provides a separate room for each scheduled meeting. In each *Meeting Room*, users may see these viewer tabs:

- Application Sharing
- Whiteboard
- Participant List

These tabs allow users to easily switch between sharing an application, viewing the electronic whiteboard or viewing a list of all users attending a meeting.

Application sharing

In Sametime meetings, users can share live applications without requiring that all meeting attendees have the application installed on their machines. The "host" can choose to share their entire screen, a window or a frame. The next figure shows what an application looks like when it is shared as an entire window.



Input
Region
Include Cursor

Output
File - BMP
i:\data\csup...\graphics*.bmp

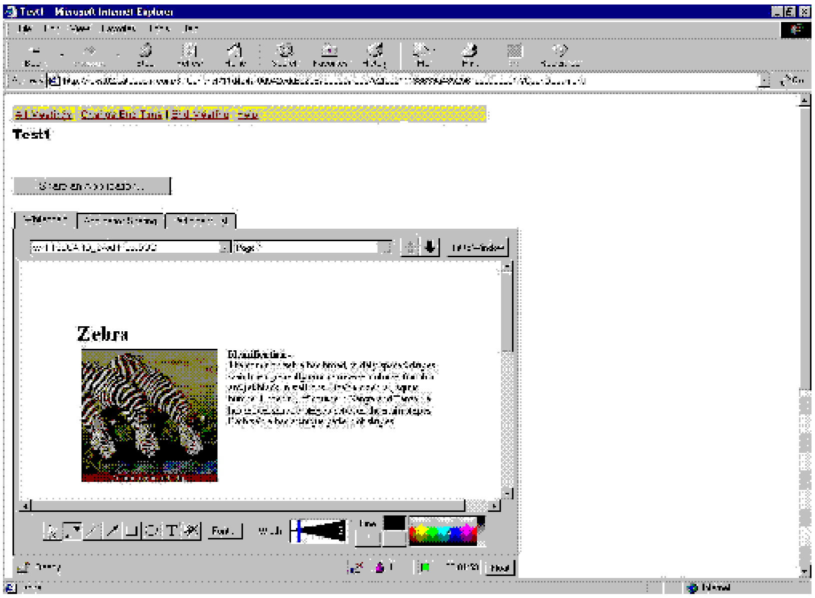
Filters
Convert to 8-bit

Options
[Preview]

Press <CTRL><SHIFT><P> to capture

Whiteboard

Examples of data that can be shared on the Whiteboard include presentations or other graphics. The Whiteboard includes annotation tools that can be used to draw on or mark up documents during a meeting.



Test1 - Microsoft Internet Explorer

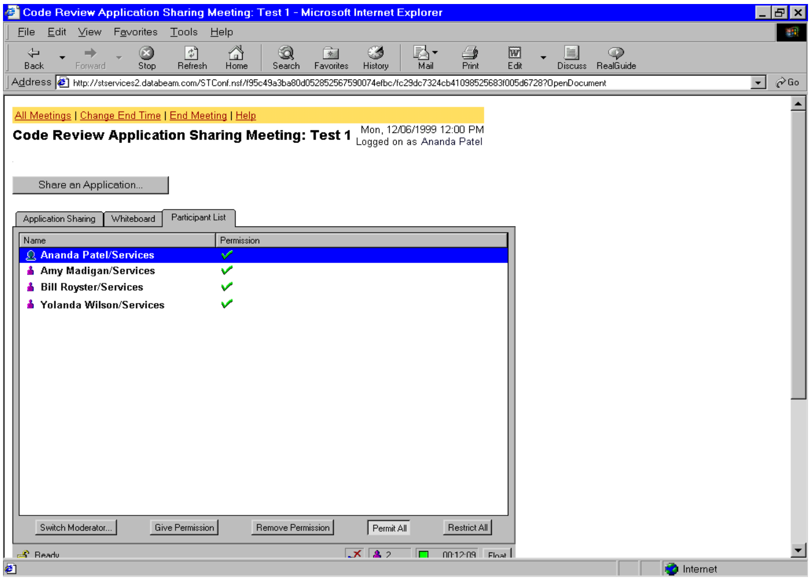
Test1

Zebra

Classification

Participant list

The Participant List tab displays the names of all meeting attendees. With the Participant list displayed, the Meeting Moderator (the one who is controlling the meeting) can permit or restrict attendees from actively participating in the meeting.



Code Review Application Sharing Meeting: Test 1 - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss RealGuide

Address http://stservices2.databeam.com/STConf.nsf/f95c49a3ba80d05285256759 0074efbc/fc29dc7324cb41098525683f005d6728?OpenDocument

All Meetings | Change End Time | End Meeting | Help

Code Review Application Sharing Meeting: Test 1

Mon, 12/06/1999 12:00 PM
Logged on as Ananda Patel

Share an Application...

Application Sharing | Whiteboard | Participant List

Name	Permission
Ananda Patel/Services	✓
Amy Madigan/Services	✓
Bill Royster/Services	✓
Yolanda Wilson/Services	✓

Switch Moderator... Give Permission Remove Permission Permit All Restrict All

Ready 2 00:12:09 Float

Internet

Sametime's Meeting Services toolkits can be used to embed meeting control, electronic whiteboarding and application sharing into any environment that supports Java Virtual Machine R1.1.4 or later. These tabs can be combined or separated out, depending on the requirements of the application.

Sametime-enabled databases

The Sametime server includes Sametime discussion, sample e-mail and sample document library templates. These templates blend online awareness and real-time chat capabilities into commonly used Lotus Domino databases.

The discussion template (STDISCUSS.NTF) can be used like any other Domino template to create new discussion databases or replace the design of existing discussion databases on Domino servers. The sample document library (STDOCLIB.NSF) and e-mail (STMAIL.NSF and STMAILW.NSF) databases are available for demonstration purposes. Developers can use these sample databases to learn how to add Sametime functionality to Domino applications.

Sametime-enabled databases can be deployed on a Domino server that includes Sametime, or on a dedicated Sametime server that functions as part of a Domino domain. The databases contain Who is Online and Who is Here buttons that display Awareness windows when clicked. The Who is Online window displays a list of users who are logged into the Sametime community.

The Who is Here window allows users to get a dynamic view of who is in a particular place with them. A place may be a database, a document, an e-mail message, and so on. A user viewing a document in a Sametime discussion database can select the Who is Here option to see a list of online users who have opened the same document in the database. Using the Join Chat Here option, the user can then engage other online users in a real-time discussion of the document they are viewing.

Application development toolkits

Two application development toolkits are shipped with Sametime. These toolkits, packaged as LotusScript and Java, can be used to Sametime-enable existing applications or to integrate Sametime into the design of new applications. The LotusScript Toolkit contains the Community Services API. The Java Toolkit contains both Community Services and Meeting Services APIs. The Sametime toolkits make it easy to integrate real-time services into any Web or Notes-based application. By integrating Sametime into applications, developers provide a single, collaborative application with all the tools necessary for users to communicate quickly and effectively.

Developer benefits

By using the LotusScript and Java toolkits to Sametime-enable applications, developers can experience these benefits:

- Reduced development time: developers will find these easy-to-use toolkits a great starting point for adding Sametime's capabilities.
- Leverage the Notes/Domino experience: Sametime's toolkits leverage the capabilities that Domino developers have already mastered, such as Domino APIs' LSX, Java modules, and Domino Designer.
- A variety of entry points: Sametime offers a wide variety of development entry points, such as embeddable Java applets, Domino C and C++ APIs, LSX modules and Domino database templates.

The Application Development Toolkits is discussed further in Chapter 3.

Server administration tool

The Sametime administration tool permits server management via a built-in Web management portal on the server. Using Java applets, the portal provides real-time statistics monitoring, forms for user registration and system configuration aids. Up-to-the-second information about Meeting Services, Community Services, the HTTP server, and disk space on the Sametime server is available in an easy-to-read format. The server administration tool can be accessed after the Sametime server has been installed.

"Fit for Business" features of Sametime

Sametime is billed as a "Fit for Business" solution because it is designed to dramatically advance business collaboration using real-time technologies. The Sametime solution offers security, privacy, scalability and proxy support. Additionally, the product is standards-based.

Security

The Sametime server includes many of the same security features as a Domino server. RC2-40 is used to encrypt meeting data (T.120 protocol for application sharing and whiteboard) and awareness data (VP protocol for instant messaging). When meetings are accessed by a Lotus Notes client, Notes provides encryption using RSA.

Privacy

Privacy can be controlled at several different levels. In Connect, Awareness can be managed so that only select people can see that you are online. User authentication ensures that the person you are interacting with online is truly who they say they are.

In the Online Meeting Center, privacy can be managed by:

- *Password protecting meetings.* In this case, only invitees will receive the password and be admitted to the meeting.
- *Creating the meeting as an unlisted meeting*. Unlisted meetings do not appear in the Meeting Center. Therefore, only the invitees know that the meeting exists. To enter an unlisted meeting, invitees must supply the exact meeting name and optionally a password. Both of these keys are provided to invitees by the meeting convener before the meeting.
- *Providing a list of people who are allowed to attend the meeting.* This is done during the scheduling process.

Scalability
The Sametime client/server architecture supports the needs of large enterprises. A single Sametime server can support tens of thousands of connected users for awareness and instant messaging. Hundreds of users can simultaneously view the same application or presentation whiteboard in an online meeting.

To increase connection capacity, you can cascade multiple Sametime servers across wide area networks and balance the load for large conferences. This setup provides local access while reducing network traffic.

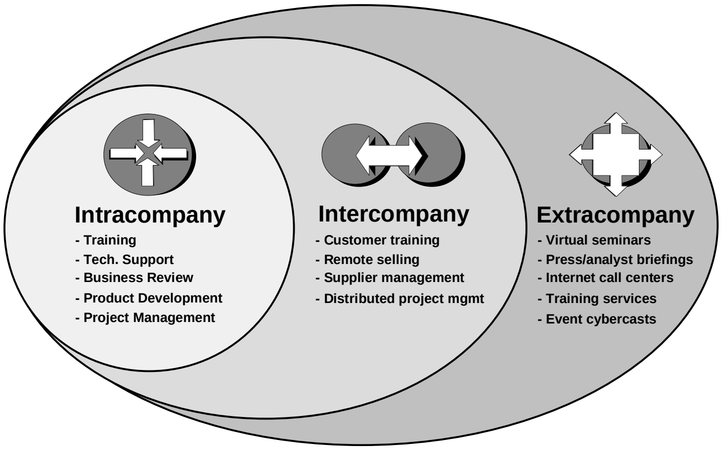
Proxy Support
Sametime provides proxy support for the Connect client, the Instant Meeting Share an Application component and the Application Sharing and Whiteboard applets of the Online Meeting Center. For Sametime Connect users, HTTP, HTTPS, SOCKS4 and SOCKS5 proxies are supported. Clients with HTTP and SOCKS4 proxies can join online meetings and share applications. Depending on the setup, Sametime can also work with SOCKS5 proxies. Application Sharing now supports proxies and tunneling.

Standards-based
For data sharing, Sametime uses the T.120 standards developed by the International Telecommunications Union (ITU). Lotus is a member of and participates in the interoperability testing events sponsored by the International Multimedia Teleconferencing Consortium (IMTC).

Sametime application examples

The Sametime target market includes enterprise businesses, government agencies and service providers. Any organization with a dispersed work force or a need to work with remote colleagues or partners can benefit by using Sametime.

The Sametime application development platform allows customers to create customized solutions to meet their specific needs. Sametime-enabled applications can be classified as intracompany, intercompany and extracompany solutions.



Intracompany solutions

Increased worker productivity can be one result when Sametime is deployed within a company to improve internal communication. When ideas can be exchanged instantly, collaboration and information sharing moves to a new level. These are ways in which Sametime can be used as an intracompany solution to bridge the communication gap for dispersed workers:

- *Virtual meetings.* With its Online Meeting Center, Sametime provides an easy way to get company employees together. Without a massive investment in new technology, Sametime provides an easy-to-use internal virtual conferencing center.
- *Technical support.* Internal help desks are a necessity in today's technology business arena. By building an electronic help desk application and adding Sametime capabilities to it, companies can free their desk agents' eternal ties to the telephone. The new application would allow the agents to answer client questions online using chat. They could also start an instant meeting and take control of the client's application to resolve an issue.
- *Product development.* All of the Sametime capabilities lend themselves to collaborative product development, especially in a Notes environment. In such an environment, the product development track can start with the creation of a discussion database, which is then Sametime-enabled. Development team members can then meet in this database to engage in chat sessions or instant meetings with others in the database.

Intercompany solutions

Sametime can be integrated with existing customer communication activities to provide intercompany solutions, for example:

- *Supplier management.* Because the Internet makes Sametime accessible to external suppliers, company purchasing representatives can easily communicate with these entities. With a Sametime-enabled Web site, purchasing representatives can chat with and share documents with suppliers. This easy communication venue can increase the frequency of contact, which will lead to better relations with suppliers.
- *Customer training.* In many cases, completing a product sale to a customer is not the end of the relationship with that customer. Many products require follow-up assistance such as installation and training. Using Sametime, companies can provide this follow-up care by conducting virtual seminars and online demonstrations.

Extracompany solutions

Extracompany solutions best demonstrate the ability of Sametime to make a difference in the way a company conducts business. The following sample solutions could also make a huge difference in the cost of doing business:

- *Virtual interviews*. Using Sametime, companies can shift some of their recruiting activities from the physical realm to the virtual realm. For example, a recruitment company could integrate Sametime into its Web site, enabling recruiters to interview job applicants online. The two parties can dialog using text chat while the candidate's resume is being displayed on the Sametime electronic whiteboard.
- *Sales support*. Again, by Sametime-enabling a company's Web site, sales representatives can provide product assistance to customers. When a customer visits a Sametime-enabled site, online sales representatives can direct them to products and answer questions about the product. The representatives can also direct them to other company product sites which could result in additional sales.

What's ahead

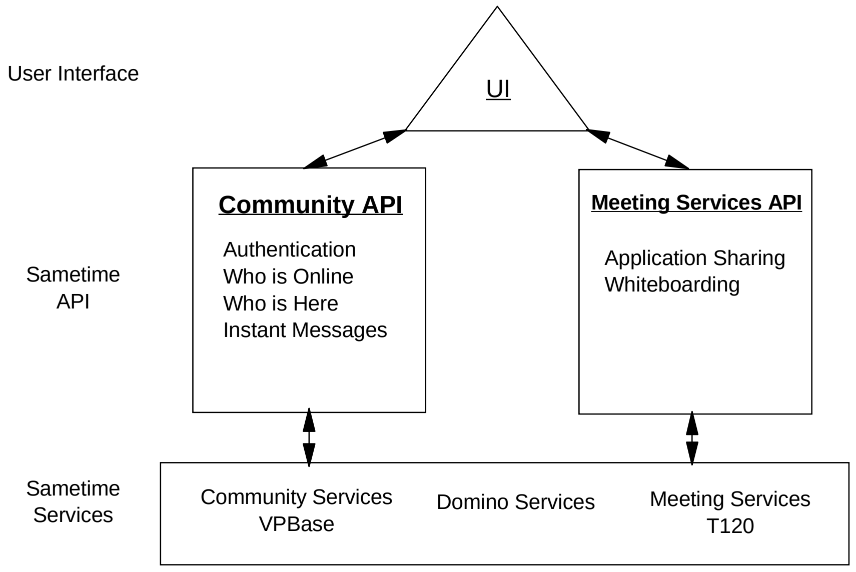
The following chapters of this book will explain the Sametime application development toolkits and how you, an application developer, can use them to enhance existing applications or to integrate Sametime's capabilities into the design of new applications. After discussing the LotusScript and Java API in detail, we show you the actual procedures involved in adding Sametime capabilities to Lotus Notes and Domino applications. The book ends with a comprehensive case study which demonstrates an actual Sametime-enabled application.

Chapter 2
The Sametime environment

In this chapter we examine the Sametime environment and show how it relates to applications. We describe the services that comprise the Sametime environment, and look at the security factors that an application developer needs to know when Sametime enabling databases. Finally, we look at the considerations for deploying a Sametime-enabled application within a production environment.

Sametime services

The Sametime server provides a set of server tasks which enable and coordinate synchronous online communication between multiple sets of users. These services can be grouped into two different categories: meeting services and community services. The remainder of this section provides a description of what these services comprise.



Community services

The community services are those which handle the interaction between users. In addition to authentication, they are also the ones which handle the chat components.

Authentication

The authentication services handle the global functions of the community services. The basic piece is the login/logout. In addition, this also includes name resolution and user capabilities. User capabilities includes setting status and setting privacy.

Login/logout

The primary authentication service is the logging in and logging out of users. When users log in, they are authenticated using their person record's full name field in the Domino directory on the Sametime server. For a password, they use the Internet password from their person document.

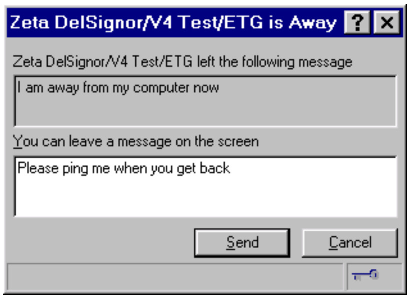
Note Users always authenticate against a Domino directory on a Sametime server, regardless of how the Sametime environment is set up.

Name resolution

The authentication service also handles the resolution of names. That is, you provide a name or list of names, and the service will return the subset of those names that are valid. The service does this by determining if each entry has a valid person record entry in the Domino directory on the Sametime server. If there is more than one entry for an individual, you can indicate how to resolve the name. If an entry is valid, it will show up on a resolved names list. If an entry is not valid, you will never see it. Names are resolved by using the API function ResolveNames with a list of names as a parameter, which is sent to the Sametime server. What is returned is the list of names in the Notes heirarchical form that are valid.

Setting status

When a user logs in to the Sametime server, the user is authenticated with the server and is placed in a state as being "Online". The user can then choose to set either of two additional statuses. One status is that the user is away, which indicates that they are on line but away from their desk. If anyone tries to send an instant message, the message will appear on their machine, but since they are away from their desk, they will not respond. The sender is notified that the user is away but they can leave a message. The following figure shows the warning message that the user receives when sending a message to someone whose status is "I am Away".


Zeta DelSignor/V4 Test/ETG is Away

Zeta DelSignor/V4 Test/ETG left the following message

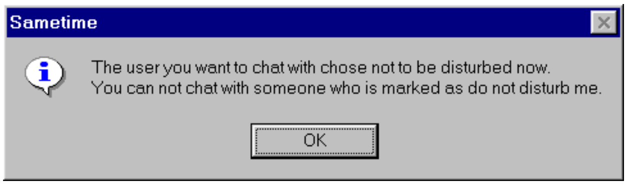
I am away from my computer now

You can leave a message on the screen

Please ping me when you get back

Note Any message a sender leaves will be unprotected. Therefore, it is important that you consider this before leaving a message on someone's screen, as it may be visible to others nearby.

The other status is "Do Not Disturb". With this status, the user is indicating that they are at their desk and using their machine, but do not want to be interrupted. Anyone trying to send them an instant message will be prevented from doing so. The following figure shows the message the user receives if they try to send a message to someone who has set their status to "Do not Disturb".


Sametime

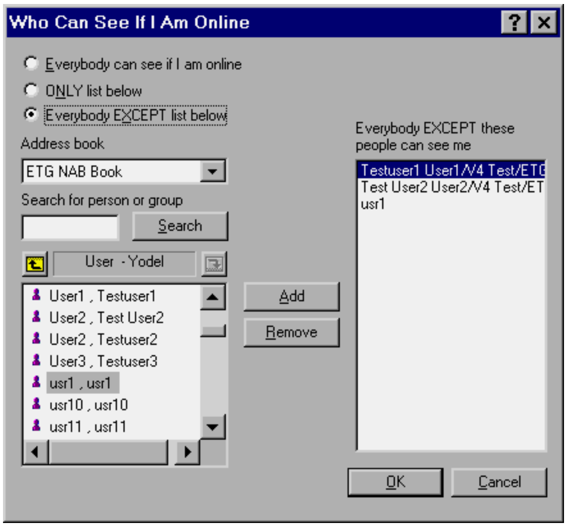
The user you want to chat with chose not to be disturbed now.
You can not chat with someone who is marked as do not disturb me.

Setting privacy

Users have the ability to specify which colleagues can see that they are logged in to the Sametime server. One reason for this is so that high-profile people, such as a company executive, for example, will have the ability to use the Sametime features, but will not get bombarded with unwanted messages. In setting a privacy list, users have three options:

- Let everyone in the Sametime community be able to see them.
- Include a list of people they want to see them on line. For example, a manager may want to only include a list of direct reports and those to whom they report.
- Provide a list of people they do *not* want to see them.

The next figure shows the privacy dialog box.



Everybody can see if I am online
ONLY list below
[✓ Everybody EXCEPT list below]

Address book: ETG NAB Book

Search for person or group

User - Yodel

User1 , Testuser1
User2 , Test User2
User2 , Testuser2
User3 , Testuser3
usr1 , usr1
usr10 , usr10
usr11 , usr11

Everybody EXCEPT these people can see me

Testuser1 User1/V4 Test/ET
Test User2 User2/V4 Test/ET
usr1

Users can set these options through the Connect client. Also, there are API functions available in the toolkit with which a developer can allow a user to set or reset these options within an application.

Important Setting or resetting the list of people that a user can see when they are online is a global function. If a privacy list is set in the Connect client, the list will also be enforced within an application and conversely, if the privacy list is set within an application, it will also be reflected when using the Connect client. This is done to prevent "lurking" either within the Connect client or within an application. Lurking is when someone can see you but you can not see them.

Instant messaging

Instant messaging is the actual server service which handles information sent and received between groups of people. This is the service that handles the actual chat messages being sent back and forth. The messages that can be sent are text (IMSendText())or data (IMSendData). In the case of a simple chat this would be simple text. Most of the instant messaging is handled through the user interface controls. However, there are API calls for the developer to control sending and receiving of messages.

Who is online

The "Who is Online" service provides an awareness concept that is *people-based*. This means that you have successfully logged into the Sametime server, and this is your status, and these are the people whom you can see. The statuses are "I'm Available", "I'm Away", and "Do Not Disturb".

Application developers have access to an API function called “OpenWhoIsOnline” where you provide the server a list of names and it returns the names of those whom you can see and what their status is. The basic question you are asking the server is: “Given this list of names, who is online and what is their status?” There are also a couple of considerations.

First, the names that are fed to the Sametime server must be valid names in the Domino Directory. Invalid names do not show up in the list that is returned by the server. The first thing the “Who is online” API function does is resolve the list of names. It verifies that each name requested is a valid name in the Domino directory on the Sametime server. If a name is not valid it will not show up in the resulting list.

Note Users always authenticate against a Domino directory on a Sametime server, regardless of how the Sametime environment is set up.

Second, we also have to include the privacy lists of the users and the names that are queried. If you have placed restrictions on who you can see, and/or any of the persons requested have placed restrictions on who can see them, these names are shown as being off line.

Who is Here

The “Who Is Here” service provides an awareness concept that is *place-based*. You are announcing to the world “I am here”. That *here* can be any place you want it to be. From the Sametime server ’s perspective, a *place* is a character string set by a user. When the “Who is here” service is queried, a character string is sent to the service and the service returns all names which have the same character string associated with them.

Think of a place as a database. Let’s use a Domino discussion database for example. When you open the database, you can identify the place to be the database ID. Anyone else who has opened the database has also entered the same place. If anyone queries “Who is here” in the database, all the names of all the people in the database will appear.

If you now open a document in the database, you enter a new place; this would be indicated by the document ID. Other people who open the same document will also be entered into this same place, and querying “Who is here” in the document will produce a list of people reading the same document.

You can be in more than one place at a time. When you enter the discussion database, you are entering one place. When you open a document in the database, you are entering a second place, but you are not leaving the first place. If you open additional documents in the database, you are entering additional places and you can see who is also in those places as well.

These places do not have to be restricted to database or document IDs. They can be any character string the developer chooses to call them. This feature can be used very creatively by the developer to enhance business processes. For example, consider the case of a help desk database, accessed by both customers and help desk agents. When the customers enter the database, they can enter a place called something such as “CustomerQueue” and the help desk agents can enter a different place from the same database, called something like “AgentQueue”. The agents can find out who is in the “CustomerQueue” and help the first one in line.

You have to be careful with naming conventions for places. For instance, there could be more than one help desk in an organization. If several help desk applications used the same names for customer and agent queues there could be some confusion and agents could potentially be trying to help the wrong people. The who is here service is not database-specific. It does not know from which databases users entered the places.

There may be some applications where defining the same place in different databases would be a benefit. For example, you may have an application with multiple databases, each having completely different designs, and you may want to know who is in the application rather than who is in a particular database. The definition of a Place in the Who is Here service gives you the flexibility to do this.

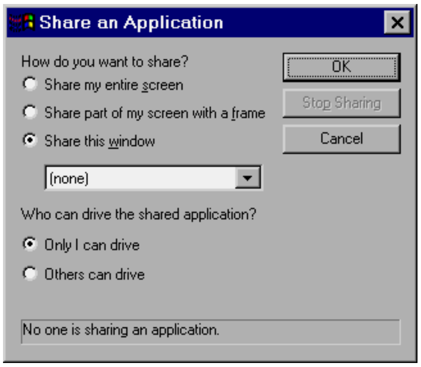
Meeting services

Meeting services allow for the synchronous sharing of data and applications. There are two basic components of this service: application sharing and whiteboarding. This section describes what each of these components does.

Application sharing

Application sharing allows users to control and manipulate applications and machines. One user can show and demonstrate an application on their machine, or they can allow multiple users to see and control applications on their machine. Meeting services handles these controls. Presently, only Windows 32-bit applications can be shared. When sharing an application, the users have two basic options from which to choose: what is to be shown and who can control what is being seen.

The next figure shows the dialog box where the user specifies these options. The choices for what is to be shown are Entire Screen, Window, or open application. The control choices are Only I can drive or Others can drive.



How do you want to share?
Share my entire screen
Share part of my screen with a frame
[✓ Share this window]
(none)

Who can drive the shared application?
[✓ Only I can drive]
Others can drive

No one is sharing an application.

What to share

The first three choices in the dialog box define what is shared with other meeting participants. The meaning of the choices is as follows:

- Share my entire screen
  This allows the user to show everything that is on their machine. If they want to move from multiple windows, or show multiple applications, or an application which opens many windows, they can select this option.
- Share part of my screen with a frame
  This allows others to see only part of what is on a user’s machine. The user can define how large a frame they want to share, and others can only see what is within the specified frame. This is useful, for example, if you want to perform a demonstration and at the same time do something behind the scenes that you do not want other users to see. This option also has the best performance for application sharing.
- Share this window
  This allows a user to share an application which is currently running on their desktop. The user first starts the application, then goes to the application sharing dialog and selects this option. Others will see only this application and nothing else running on the controlling user’s desktop. This choice allows the person who is sharing this application to run another application simultaneously, without other participants in the meeting seeing the other application.

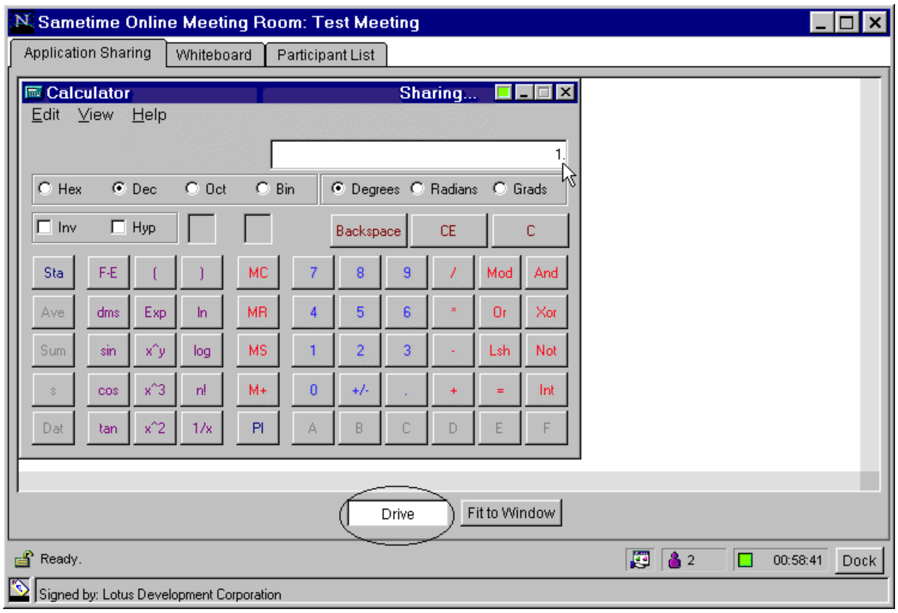
Control of shared applications

Once a user selects what they want to share, they can choose who can control the shared items.

- Only I can drive
  This is the default option; it means that only the person who is sharing the application can control it. Other participants in the meeting can only watch what's going on.

- Others can drive
  This option allows participants other than the user who has shared the application to actually control it. When this choice is selected, users click on the Drive button on the applet and then use their own machine to drive the application.

The following figure shows an application being shared, and a participant controlling the application.



Sametime Online Meeting Room: Test Meeting

Calculator — Sharing...

Display: 1.

[● Dec] Hex Oct Bin — [● Degrees] Radians Grads

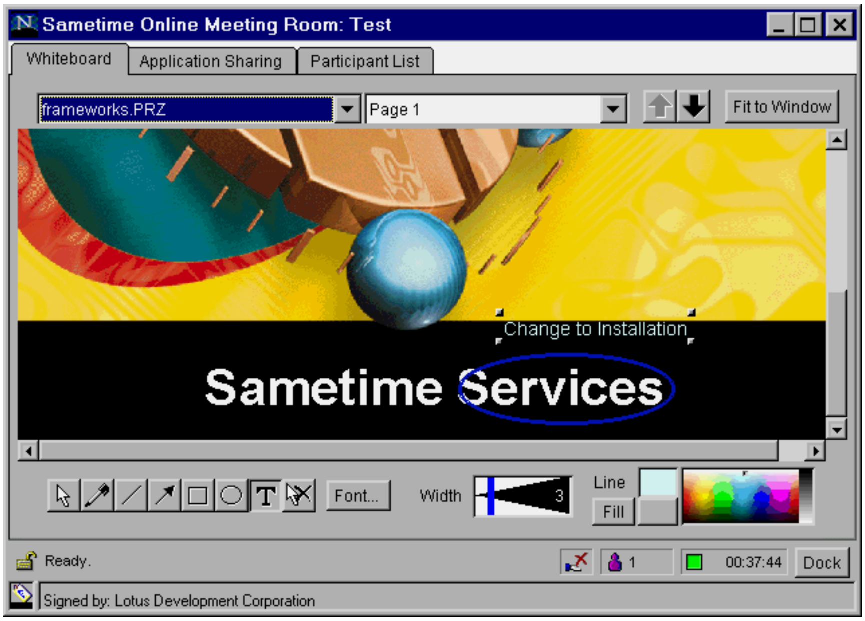
Ready.

Signed by: Lotus Development Corporation

The participant pushes the "Drive" button (circled), after which they can manipulate the application (here typing in a "1").

Whiteboarding

This feature allows for use of an electronic whiteboard, which lets participants draw, mark up and scribble on the electronic version just like they would on its physical counterpart. The whiteboard in Sametime also allows an application to be loaded behind it. Participants can mark up and comment on the application without affecting the underlying application itself. For instance, a group of people may be reviewing a Freelance presentation, and individuals in the group may draw out comments on the presentation, as illustrated in the next figure.



There is one limitation to this feature, at least at present: there is no formal way that data on the whiteboard can be saved at the meeting's conclusion. The users can press the Print Screen button to capture the data, then load it into a program such as Paint. However, users can not reload the data into a new meeting to continue a discussion.

The API functions involve calling a single Java applet which handles both the application sharing and the whiteboarding. The API functions allow for the loading and manipulation of the applet. The intent was to give the users all the available tools with which to conduct a meeting and not place restrictions on what the moderator or participants can do. Otherwise, one would have to specify which functions would be needed before creating the meeting, and there would be no flexibility to change in the middle of a meeting.

Sametime application security

There are two levels of security in a Sametime-enabled database: authentication with the application itself and authentication with the Sametime server. When we enter an application, we first authenticate with the application database. Once that authentication is complete, we then go outside the application database to the Sametime server and authenticate with it. This is done regardless of whether or not the application is deployed on the Sametime server.

If we are using a Notes client rather than a browser, application security is handled by the Notes security mechanism. When accessing the database using a Notes client, the Notes ID mechanism is used for authentication. When accessing the database using a Web browser, the user's name and internet password stored in the Domino directory person record are used.

The second authentication is with the Sametime server. This is a separate authentication mechanism. There are two different means of authentication, which are described in the next sections.

Passwords

One option developers can use is to require users to log into the Sametime server with a password after being authenticated by the application. Use the "loginbypassword" API function to do this. When this method is specified, users must use the internet password set in their person document in the Domino directory. From a usability standpoint, this is not recommended. Every time a user enters a database, they will have to type in their user name and password to log into the Sametime Server; repeated logging in may become annoying to users.

Secrets and Tokens

There is a mechanism with which users who are already authenticated by an application can seamlessly log into the Sametime server without having to actively type in a second password. Two databases, Secrets (StAuthS.nsf) and Tokens (StAuthT.nsf), are created when the Sametime server is installed; they are replicated to application databases on which Sametime-enabled applications are hosted. These databases must be placed on the main Sametime server data directory with the same file name indicated above. Script library code placed in Sametime-enabled applications looks for these specific file names. The agents used to log into the Sametime server have the source code removed and reference these databases by name, so this cannot be changed.

Secrets

The Secrets database contains keys used in generating tokens. These keys are updated daily. It has the file name "StAuthS.nsf and must be placed in the Sametime server's data directory. Because the Secrets database is used in the authentication scheme, direct access should be strictly controlled. Granting access to a large number of users would compromise your ability to maintain system security. There are two agents that must be added to a Sametime-enabled database. They are (SametimeQueryOpen) for databases that are accessed using a Notes client and (SametimeWebQueryOpen) for databases accessed using a browser. They are located in the Sametime Toolkit Utilities database (StToolkitUtils.nsf) on the Sametime server. Copy these agents from the Toolkit Utilities database into the application.

Important These agents are used to directly access the Secrets database and must be signed with a Notes ID (either person or server) which has access to the Secrets database. The source code is not available in these agents, so users cannot trace it and therefore determine the identity of the secrets.

Tokens

The tokens database is used to process authentication requests from users. It serves as a temporary placeholder for user tokens and contains documents indicating the nearest Sametime server where users will attempt to log in. We first invoke a LotusScript function called SametimeProfileGetToken(). Through this function we can read the Secrets database and generate the token. We then use a "loginbytoken(token)" API function and feed it the token to log in to the Sametime server. In this function, we pass it the token generated by the LotusScript function. The Secrets and Tokens databases with the appropriate function calls are used to log into the Sametime server only.

Note The application developer has to decide on the authentication mechanism for access to the Sametime server. Some of the existing applets, such as the Meeting services applet, require a particular authentication mechanism, so the decision to use them forces your decision on authentication. The recommended mechanism would be to use the secrets and tokens databases for authentication. This provides seamless integration between the application and the Sametime server.

Overview of the Sametime environment

In this section we describe the possible development environments and what features are available in each. A Sametime server can be installed in either of three different scenarios, as follows:

New Sametime server in a Domino environment

A Sametime server can be installed in an existing Domino environment as a new server. In this case the developer has all the capabilities of Domino along with the capability to add Sametime features. They also have the ability to separate the application from the Sametime server, where the application is hosted on a server separate from the Sametime server.

Installed on top of an existing Domino server

A Sametime server can be installed as a new NT service on top of an existing Domino server. (As of this writing this is only supported for an existing R4.6 server). Again, application developers have all the capabilities of Domino along with the capability to add Sametime features. This can be done by installations where resources are limited and additional hardware may not be an option.

Stand-alone Sametime server

The Sametime server can be installed as a stand-alone Web server. This allows installations which do not have a Domino environment to install and use Sametime. In this case, applications are developed for Web-only solutions. Developers can create applications using the features that Domino provides or they can build applets using a third-party applet environment and then host them on the Sametime server.

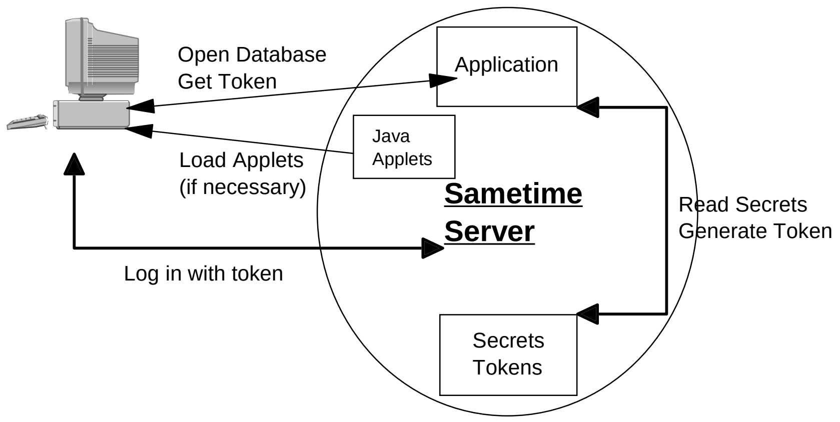
Important To a Notes and Domino developer, the Sametime server has the same look and feel as any Notes and Domino environments. They have access to the same features and function sets as they would in a normal Domino environment.

Deploying Sametime-enabled applications

In this section we look at the two basic scenarios in which to deploy Sametime-enabled applications. Applications can be deployed directly on the Sametime server, or they can be deployed on a separate Domino application server. Some organizations may already have deployment policies established. For those organizations that need to define a deployment strategy, we examine the advantages and disadvantages of both methods.

Directly on a Sametime server

The first option for deployment is that you can place a Sametime-enabled application directly on the Sametime Server. The following graphic shows an application installed directly on a Sametime Server.



The advantages and disadvantages of this method of deployment are described in the next sections. If the Sametime server is in a non-Domino environment, this is the only option available to the developer.

Advantages

The biggest advantage is that we are not critically dependent on replication of the *Secrets* database. Usually, the secrets database must replicate to servers which host Sametime-enabled applications. This replication must happen in a timely manner because the documents in the secrets database are updated daily. Some installations may require even more frequent updates. If replication is broken, then users cannot access the Sametime server through the application. With the application installed directly on the Sametime server, replication is not necessary. Even with the Domino application directly on the Sametime server, the secrets and tokens databases must be on the server with the proper access levels.

Disadvantages

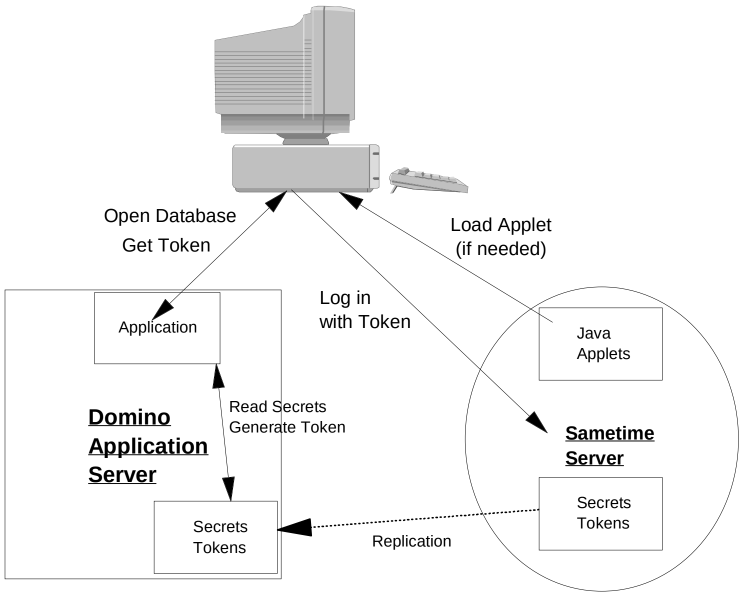
Any application running on the Sametime server is competing with the Sametime services for computer resources. You have all the various Sametime-related services on the server. Each requires CPU, disk and memory space. If you add an application to the server, the application database requires additional CPU, disk and memory space. In this case, the application is competing for the same resources as the Sametime services. This could cause performance problems for both the Sametime services and for the applications.

Another disadvantage of placing an application directly on the Sametime server is that you will not be able to use all the features available in Domino. This applies to Sametime servers that are installed as a separate server in a

Domino environment. In this case, only the basic features of Domino are installed. None of the advanced features, such as clustering or Domino Enterprise Connection Services (DECS), are available. For Sametime servers installed on top of an existing Domino server, this is not a problem. In this case, all the advanced Domino features that were installed on the server are still available.

On an existing Domino server

You can install a Sametime-enabled application on a Domino server that is separate from the Sametime server. You can have all the advantages that a Domino server has available with the additional advantages of having Sametime features. In this section we describe how you can do this, and the critical pieces that need to be in place in order for it to work. We also describe the advantages and disadvantages of placing the application on a separate server. The following graphic shows an application installed on a separate Domino application server.



Advantages

The biggest advantage is that you have dedicated resources for running the application. CPU, memory, and disk space resources for the application are not competing for resources with the Sametime services. You can also take advantage of the advanced Domino feature set. If the Sametime server is installed as a separate server, it only installs the basic features of Domino. By using a separate application server, you can take advantage of the advanced features that can be installed on a Domino server, including all the new features in Domino Release 5. These features in the new release will also be available to the developers.

Issues to Consider

There is a critical dependency on replication of the secrets database to allow users to log in to the Sametime server. If users cannot log in to the Sametime server, none of the Sametime features will work. Replicas of the secrets and tokens databases must be placed on every Domino Server that is to host Sametime-enabled applications and replication must happen regularly. If the replication between the Sametime server and the application Domino server breaks, users will not be able to log in to the Sametime server and use the Sametime features.

Another issue is that there is a dependency on having two servers up and running. Because you have two servers there are more points for failure. Both servers must be up and running and the network communication between the two must also be working for the application to be usable.

Sametime-enabling an application server

To Sametime-enable an application server, you must replicate the secrets and tokens databases onto the existing Domino server. Once you do this, the server is able to host Sametime applications. A replication schedule should be set up so that the secrets database is replicated from the Sametime server to the Domino server at least once a day.

The tokens database can be a copy, provided that the three documents pointing to the Sametime server are in the tokens database and are correct. The three documents are:

- SAMETIMESERVER - This contains the name of the Sametime server.
- SAMETIMESERVERHTTPPORT - This contains the HTTP port number.
- SAMETIMESERVERIP - This is the internet name of the Sametime server.

If you copy the tokens database to the application database rather than pull a replica, and you change the Sametime server users log into, you need to edit these three documents on each server rather than replicate the changes out from a Sametime server.

You must also set up access to the tokens database so users only see the server pointers and only their own tokens. Otherwise, they could log in as someone else.

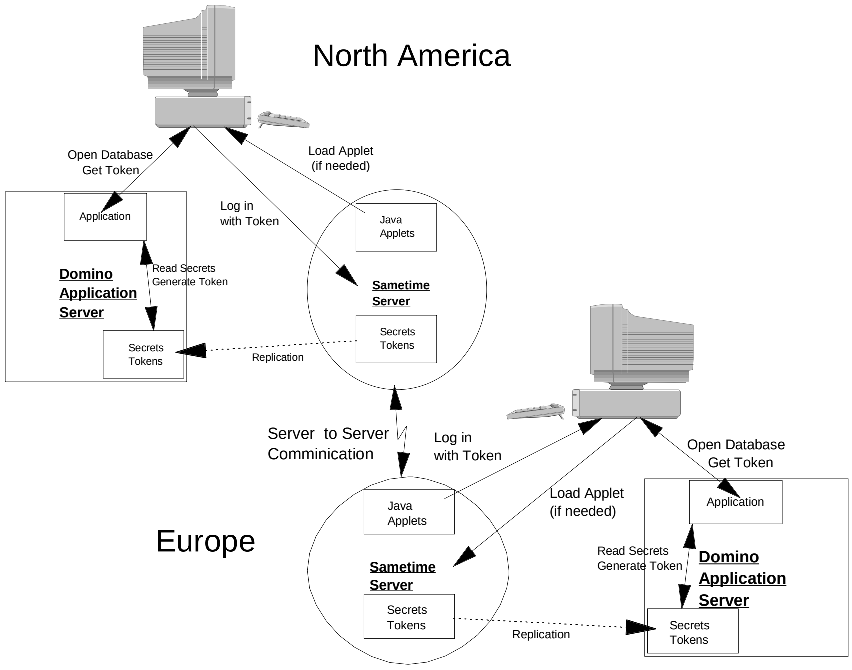
Storing Java applets

In Domino Release 5 applications, developers have the ability to store applets and code inside their application database. For Sametime-enabled applications where Java applets are used and the application is accessed via a Notes client only, developers can take advantage of this. If the application is accessed through a Web browser, the applets must be stored on the Sametime server file system. In this case you load applets from the Sametime server into your browser, then log in to Sametime.

Note You cannot store the applets in the application on the Domino server and use them to log in to the Sametime server when you access the application through a Web browser. In this instance you would be loading the applet from the application server and would then try to use this applet to log in to the Sametime server. This scenario is not supported in the Java applet security model. You have to store the applets on the Sametime server in this instance. This way you are loading the applet from the Sametime server and using it to connect back to the Sametime server — a model which *is* supported in Java. There may be policy/procedural/testing issues within your organization involved with putting files on the Sametime server and those must also be followed.

Enterprise-wide application deployment considerations

There are Domino applications that are deployed on a worldwide basis, in which replicas of the database are hosted on different servers in different locations. More often than not, there is a local Sametime server which is installed and can communicate with other Sametime servers in the organization. (The setup and maintenance of these is beyond the scope of this publication.) In this case, the developer should consider a local setup whereby the application is hosted on a separate Domino application server. The following graphic shows a single application hosted on different application servers in different locations with Sametime servers in the different locations.



One thing to consider: if the application requires applets to be placed on the Sametime server, they must be placed on all Sametime servers to which users log in. Otherwise, in some locations the Sametime features will not work.

External application deployment considerations

In some cases, external organizations may need access to an application. Typically these are Web-based-only applications. This may involve firewalls, where applications sit either outside a firewall or as part of the firewall infrastructure. There are a whole host of combinations of application and Sametime servers where access is required both inside and outside a firewall. These are infrastructure issues which are beyond the scope of this publication, however, the basic premise still remains the same: the application is either hosted directly on a Sametime server or on a separate Domino application server.

Chapter 3
Sametime API

Using Sametime for attending meetings or instant messaging within your workgroup is only a part of what is possible with the Sametime technology. Sametime can be integrated with Lotus Notes, Domino or Web applications. Imagine incorporating chat, object sharing or awareness into your application.

Lotus has provided developers with several mechanisms to “Sametime-enable” their own applications via a set of Sametime APIs.

In this chapter, we explore the Sametime APIs and explain the different pieces and their purpose.

Sametime services

The Sametime platform offers users two different methods of real-time collaboration: community services and meeting services.

Sametime presents its functionality in two different forms. *Community* is the concept of multiple users in the same virtual place at the same time. The users have some common theme and come and go from the community as they please. A *Meeting* is an event, whereby at any particular time a group of users will assemble at a particular virtual location to collaborate on a specific topic.

Community services

Sametime introduced a new term for your Lotus vocabulary: community. A community is defined as a group of Sametime servers and users. This is very similar to the concept of a Lotus Domino domain.

A place is a virtual area within the community. Places are identified by a place name. The name could be a URL, a database replica ID or a document universal ID. Users enter and exit a place as they work with an application.

As you enter a place you are able to see other people who are also in the same place at that time. Conversely, people already in that place are also able to see you as you enter. This is the definition of awareness. Others are aware of a user’s current location within the community.

Communities can provide an interesting aspect to your applications. Think of your product catalog as a place. Customers browse your catalog over the Internet like they would browse your store in a shopping center. Your customers can see if a person that knows something about a product is available. With Sametime, your salespeople can see when their customer enters the catalog “place” of your Web site. In addition, the customer would see if their salesperson or product expert is online.

Since the product catalog is Sametime-enabled, the customer can select the product expert’s name and ask questions in real-time.

Meeting services

The meeting services of Sametime allow users in multiple locations to attend a virtual meeting using a Lotus Notes client or a Web browser. Presentations or other content can be shared in the meeting so that attendees can view it. For example, you can share a Lotus Freelance presentation with others attending the meeting. A whiteboard function is also provided to allow marking up of documents or to facilitate a brainstorming session. These features are described in Chapter 1.

A unique feature of the meeting services is the ability to share an application. Attendees can view the host’s machine and applications running on it. The host can give permission to attendees to drive their machine.

Sametime 1.5 introduced *instant meetings*. From the Sametime Connect client you can select a user and send an invitation to meet. A meeting is created on the Sametime server and you can share an application and chat.

Overview of the Sametime APIs

The Sametime APIs are grouped into two functional packages. The Community Services API provides access to Sametime community services, such as instant messaging and awareness. The Meeting Services API provides access to Sametime’s whiteboard and object sharing facilities.

The Community Services API is available in three languages: an LSX for LotusScript, a set of class files for Java and libraries for C++. For this discussion, we will talk about the API in general and make note of the differences between the platforms as necessary.

The Meeting Services API is built in Java. However, many of the features are accessible via parameters that can be set in a Lotus Notes form. The applets can also be manipulated via HTML and JavaScript.

Community Services API

The Community Services API can be divided into three major parts: VP base, semantic components and user interface components. The semantic and user interface components can be further categorized by the functionality that they provide.

VP base

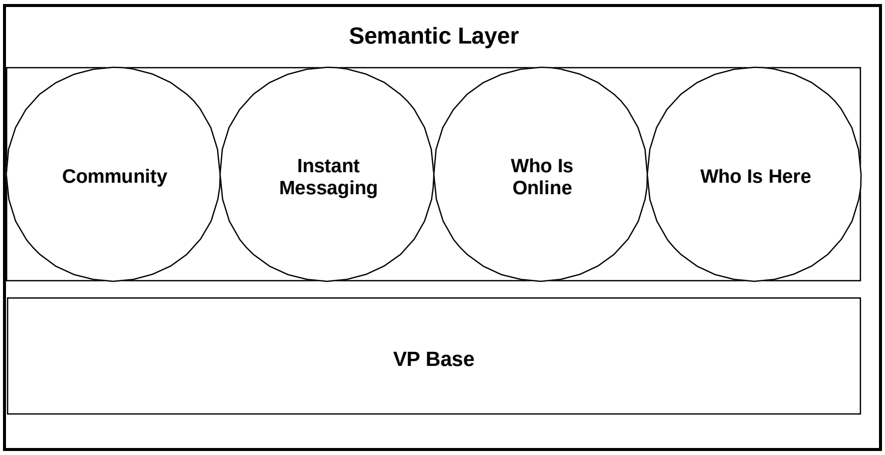
VP base is the lowest layer of the Community Services API. Its purpose is to provide other Community Services API components with access to the core VP protocol.

Note VP base cannot be accessed directly. It is an *internal* layer of the API. VP base is mentioned here just to explain the relationship of all the parts of the API.

Semantic components

To access the services of VP base, the community API offers a set of components known as *semantic components*. The purpose of these semantic components is to give access to the core functions of the Sametime server.

The semantic components of the Community Services API can be compared to the Domino back-end classes. These objects allow you to access and manipulate the services of a Sametime server, but do not provide a user interface.



This lack of a user interface should not be seen as a limitation, but as an opportunity. The semantic components contain most of the basic functionality of Sametime, and their methods could be used in the construction of your own chat client.

Community service

The community service handles all the administrative tasks of working with a Sametime community. Examples of community service operations are:

- User login and authentication
- Maintenance of user privacy settings
- Updating current user status in the community
- Resolving names to Sametime user IDs

Note The Sametime server handles users in a slightly different manner than the Lotus Domino server does. This difference will be discussed in a later section of this chapter.

Instant messaging service

The instant messaging service allows developers to create a session on the Sametime server to send and receive text and binary information via Sametime's chat facility.

Messages can be classified into different types. There are currently types defined for Sametime messages and AOL Instant Messenger messages.Developers can create their own message types, but a set has been reserved for future interoperability.

This component supports different message types to support interoperability with other vendors' products.

Important Several message types are reserved. See the API reference documentation that ships with the API for a current list of the reserved types.

WhoIsOnline service

The WhoIsOnline service provides information on the online status of a set of users within the community.

WhoIsHere service

The WhoIsHere service returns information on whether a user or a set of users is currently in the defined place. Places are virtual areas within a Sametime community. A place can be defined by a database's replica ID, a URL or any other identifier.

This service provides the ability to let the Sametime server know when a user has entered or left a place, as well as communicating with other users in the same place.

User interface components

To give a developer the ability to Sametime-enable an application without knowing all the details about the semantic components, Lotus created the user interface components.

By using the User Interface Components, developers do not need to concern themselves with the more mundane tasks of operating a chat window. Initiating a portion of the community services is as easy as calling a single function.

There are three user interface components.

Chat

The Chat component uses the community service and instant message service to provide point-to-point conversation. This component contains all the functionality that is needed to create and maintain the conversation.

Who Is Online

The Who Is Online component provides you with the functionality to determine if a particular user is logged into the community via Sametime Connect. The Who Is Online component lists the current status of a user or a group of users and allows you to start a chat with a selected user.

Who Is Here

The Who Is Here component is used to determine whether or not a user or group of users is in a particular place. As we said before, a place is a virtual area within the community that is defined by some identifier.

For example, you might develop a company discussion database for your intranet. A user could visually see if other people are in the discussion database with them. And, like the Who Is Online component, the user can initiate a chat or send a message to everyone in the same place.

Who Is Here versus Who Is Online

The difference between Who Is Here and Who Is Online is very subtle. Who Is Here answers the question “Is anyone in the same place as me?” As discussed previously, a place can be a document, a database or a section of a Web site. The list of users that is returned is in the same place and chat invitations can be sent to them or a message can be sent to the group as a whole.

An example of how this would be used might help to put these features in the correct context. When you are in a discussion database, you might want to see if anyone else is in the database looking for the same things you are. They might be able to help find the information. You click a button entitled

“Who is in this database” and a list of current users is shown. You can then select a name from the list and converse with that individual via a text chat.

An example of using Who Is Online in a Lotus Notes application would be your mail database. Your manager sends you an e-mail regarding a new product and he wants an answer as soon as possible. You click the “Who is online” button and a list of the users who are online, from among those in the To, From and CC fields of the e-mail, pops up. If your boss is online you can select his name and converse via a text chat.

Meeting Services API

The Meeting Services API allows you to integrate application sharing and whiteboarding into your application. The API was designed to create ad hoc or *instant* meetings on the Sametime server.

The API is exposed as a Java applet and can be manipulated via several parameters. These parameters can be set in the HTML <APPLET> tag, or via JavaScript function calls.

Some of the operations that you can perform with the Meeting Services API include:

- Creating an instant meeting
- Joining and leaving meetings
- Manipulating the applet’s viewers

Important Only Windows 32-bit workstations can host an application sharing session. The hosting software is installed with Sametime Connect.

We can continue the example from the Who Is Online section to explain how you can use the Meeting Services API. To answer your manager’s questions, you might want to direct him to an R&D Presentations application. This application has Sametime meeting services to enable the presentation of data across the company via the intranet. You could create a meeting and launch one of those presentations, inviting your boss. Using the text chat and the online meeting capabilities your manager gets all of his questions answered.

Working with the APIs

All the APIs are installed onto the Sametime server during the standard server installation. They are packaged into three toolkits: LotusScript, Java and C++.

Important The toolkits are constantly being improved. Be sure to check the Sametime Developers area of the Lotus Web site, **`http://stdev.lotus.com/`** to make sure that you have the most up to date version.

Programming tools

The Sametime APIs were built to allow a developer to select any development tool they wish when designing a Sametime-enabled application.

LotusScript Toolkit

Only the Community Services API is available from the LotusScript Toolkit.

The LotusScript Toolkit is made up of a LotusScript eXtention (LSX) file, some LotusScript libraries, and agents. The LSX file is installed when you install Sametime Connect client onto your workstation. The script libraries and agents can be found in the Sametime Toolkit Utilities Database.

Note The Sametime Toolkit Utilities database can be found on your Sametime server. Follow the Toolkit link from the Sametime developers home page.

Java Toolkit

The Community Services API and Meeting Services API are available in the Java Toolkit.

The Community Services API is broken up into several Java packages. The packages are archived into a CAB file for Internet Explorer and a JAR file for Netscape Navigator. The archive files can be found in the Sametime Toolkit Utilities database.

C++ Toolkit

Lotus has also packaged the Community Services API into a C++ API. With the C++ API, you can build awareness and chat into any application.

The API includes the header files and libraries required to add the Community Services functionality to your programs. Sample projects are also included.

Note At the time of this writing the C++ Toolkit was still in development. Please visit the Sametime Developers site to download the latest version.

Pre-built applications

Sametime comes with several Sametime-enabled application templates that showcase how you can use the Sametime APIs in a Lotus Notes or Domino application. You can use these templates as a starting point. You can also visit the Sametime Developers site to get more information on Sametime development, as well as updated toolkit information and sample applications.

Summary

In this chapter we explained the pieces of the Sametime API necessary to Sametime-enable an application, whether it is Notes-, Domino-, or Web-based. The Community Services and Meeting Services API each allow access to the many real-time collaboration features. The multiple toolkit platforms make selecting the appropriate development tool easy.

In the following chapters, we look at how to add each feature (Who Is Online, Who Is Here and Meeting Services) to an application.

Chapter 4
Adding Who Is Online awareness

Who Is Online is the people-based awareness service found in Sametime. A part of the Community Services API, it provides awareness of users in a Sametime community. When a client supplies a list of users about whom to determine online presence, the Sametime server displays a pop-up window showing the individuals that are online. These users could be logged in to the Sametime server via the Connect client or through another Sametime-enabled application.

In addition to showing a list of the individuals online, Who Is Online awareness provides each user's online status. This status can be any of three types: I am Active, I am Away or Do Not Disturb Me. If a user is *Active,* they are logged in to the Sametime community and available for communication. The *I am Away* status indicates that a user is online, but is currently away from their computer. When a user sets his status to *Do Not Disturb Me*, he is able to work without being disturbed by others in the Sametime community.

The Sametime toolkits can be used to add Who Is Online awareness to applications. These toolkits are available as LotusScript, Java and C++ packages.

Note At the time of this writing, the C++ Toolkit was still in development. It will not be discussed in this chapter. Please visit the Lotus Sametime Developers' Site for more information on this toolkit and to download the latest version.

Sametime toolkits

Both the *LotusScript* and *Java toolkits* contain the Community Services API. The LotusScript Community Services API contains *semantic* and *user interface* (UI) components. The semantic components provide access to the server's core functions. They expose all the functionality of Sametime and allow developers to integrate Sametime capabilities into applications in a flexible way. New user interfaces for Who Is Online capabilities can be created using Community Services' semantic components.

The Who Is Online UI component can be used by developers to easily add Sametime functionality to new or existing applications. This component provides a higher level of functionality by gathering the semantic component capabilities into a set of predefined user interface controls. Community

Services' Who Is Online UI allows for the creation of a popup window containing the list of online users and their status.

The Java API contains both Community Services and Meeting Services APIs. Like the LotusScript API, the Java Community Services API is divided into semantic and UI components. Each component is represented by a separate class.

The Java Meeting Services API is discussed in Chapter 6.

Add Who Is Online awareness to a Notes or Domino application

You can add Who Is Online awareness to a Notes or a Domino application. LotusScript or Java can be used to Sametime-enable Notes applications. Adding Who Is Online awareness to a Domino application makes extensive use of standard Web programming languages such as Java, HTML, and Javascript.

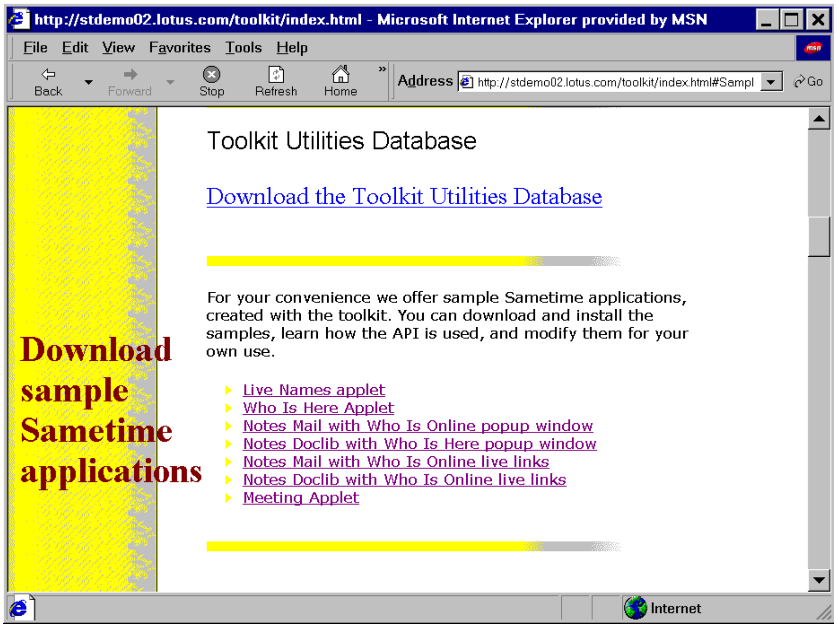
If Who Is Online awareness is added to a Notes application, the new application can be accessed using a Notes client. Domino applications that have been Sametime-enabled can be accessed with a Notes client or with a Web browser. Either Netscape version 4.0.6 or higher or Internet Explorer version 4.x is necessary to access Sametime-enabled applications via the Web.

In this chapter we describe the steps to add Who Is Online awareness to a Notes application using LotusScript and Java. We also show you how to copy the Who Is Online functionality available in the Notes 5.0.2 mail form and add it to another application.

Adding Who Is Online awareness to a Notes application using LotusScript

To demonstrate adding Who Is Online awareness using LotusScript, we will Sametime-enable a Notes mail template. When finished, users will be able to access the form and find out who is online.

When you install the Sametime server on your machine, a Welcome to Sametime Web page is included. This page contains the Sametime Client Package, the Sametime toolkits, and several sample applications. The sample application that we will use in our example, Notes Mail with a Who Is Online pop-up window, can be found on this page.


http://stdemo02.lotus.com/toolkit/index.html - Microsoft Internet Explorer provided by MSN
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home
Address http://stdemo02.lotus.com/toolkit/index.html#Sampl
Go
Internet

Toolkit Utilities Database

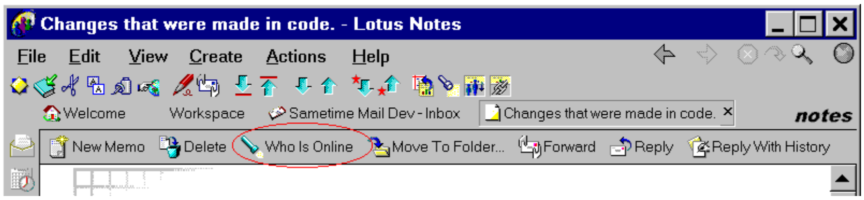
Download the Toolkit Utilities Database

Download sample Sametime applications

For your convenience we offer sample Sametime applications, created with the toolkit. You can download and install the samples, learn how the API is used, and modify them for your own use.

- Live Names applet
- Who Is Here Applet
- Notes Mail with Who Is Online popup window
- Notes Doclib with Who Is Here popup window
- Notes Mail with Who Is Online live links
- Notes Doclib with Who Is Online live links
- Meeting Applet

The sample mail application included with the Sametime server is built on the Notes 4.6 platform. We will start with a copy of the mail template and step you through the process of adding Who Is Online awareness. This functionality will show up in the action bar of the Sametime-enabled form.


Changes that were made in code. - Lotus Notes
File Edit View Create Actions Help
Welcome
Workspace
Sametime Mail Dev - Inbox
Changes that were made in code.
notes
New Memo
Delete
Who Is Online
Move To Folder...
Forward
Reply
Reply With History

Setting up the environment

This application will be hosted on a Domino R5 server with a Sametime 1.5 server in its domain. Note that users must install the Sametime Client package on their machines in order to take advantage of Who Is Online awareness in Notes applications. This Client Package can be found at the Lotus Sametime Welcome page.

Preparing the template

The *Sametime Toolkit Utilities database (STToolkitUtils.nsf)* contains all the necessary agents and scripts used to Sametime-enable an application. This database is included with the Sametime server package and can be downloaded from the Lotus Sametime Welcome page.

To prepare the mail template to embed the Who Is Online awareness, we will copy several script libraries for the Sametime Toolkit Utilities database. It is necessary to also copy and sign an agent from this database.

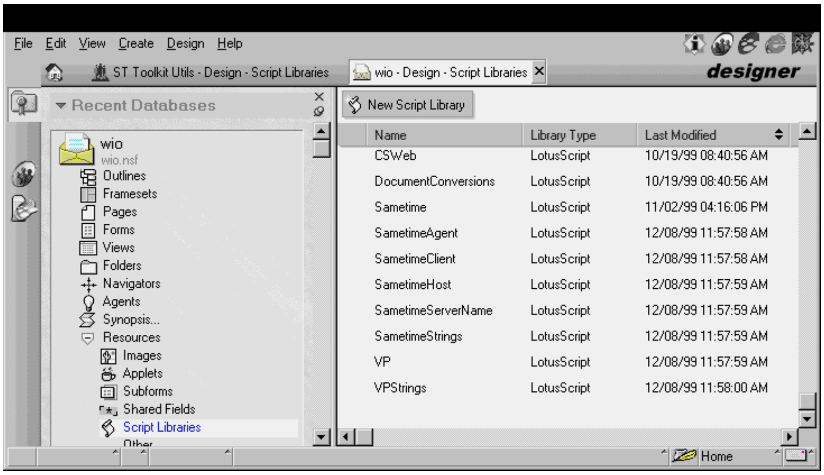
Adding the script libraries

A script library is where you store code segments that you want to use from other scriptable objects. The Sametime script libraries contain code that will locate a user's Sametime server and assist in the login process.

From the Sametime Toolkit Utilities database Script Libraries view, *copy* the script libraries listed in the following table and *paste* them into the same view in the mail template.

Script Name	*Script Function*
SametimeAgent	A script program used in the Sametime server login process. (The source code is removed from this script.)
SametimeClient	A script program used in the Sametime server login process. (The source code is removed from this script.)
SametimeHost	Scripts used in locating a person's Sametime server.
SametimeServerName	A script program used to locate the Sametime Server associated with the application.
SametimeStrings	A set of translatable strings called by the Sametime script libraries.
VP	Provides all the Virtual Places functionality. Loads the ActiveX components and handles login and logout to and from the Sametime Server.
VPStrings	A set of translatable strings called by the VP Script.

The copied script libraries will append to existing ones in the template.

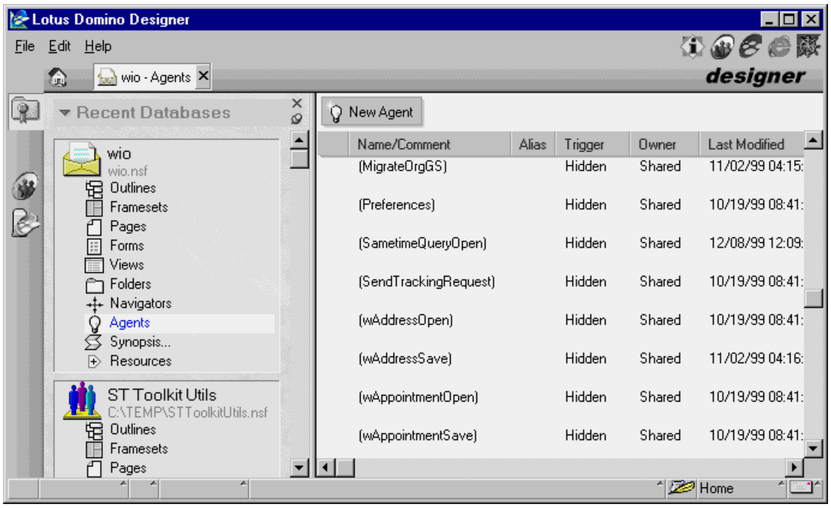


Name	Library Type	Last Modified
CSWeb	LotusScript	10/19/99 08:40:56 AM
DocumentConversions	LotusScript	10/19/99 08:40:56 AM
Sametime	LotusScript	11/02/99 04:16:06 PM
SametimeAgent	LotusScript	12/08/99 11:57:58 AM
SametimeClient	LotusScript	12/08/99 11:57:58 AM
SametimeHost	LotusScript	12/08/99 11:57:59 AM
SametimeServerName	LotusScript	12/08/99 11:57:59 AM
SametimeStrings	LotusScript	12/08/99 11:57:59 AM
VP	LotusScript	12/08/99 11:57:59 AM
VPStrings	LotusScript	12/08/99 11:58:00 AM

Add the SametimeQueryOpen agent

The *SametimeQueryOpen* agent reads the *Secrets database (STAuthS.nsf)* and generates a token from the *Tokens database (STAuthT.nsf)*. That token is then used to authenticate Notes clients and log them onto the Sametime server. The Secrets and Tokens databases are created when the Sametime server is installed. See Chapter 2 for more information on these databases.

To ensure user authentication to your Sametime-enabled application, *copy* the SametimeQueryOpen agent from the Sametime Toolkit Utilities database in the Agents view, and *paste* it into the same view of the mail template.



Name/Comment	Alias	Trigger	Owner	Last Modified
(MigrateOrgGS)		Hidden	Shared	11/02/99 04:15:
(Preferences)		Hidden	Shared	10/19/99 08:41:
(SametimeQueryOpen)		Hidden	Shared	12/08/99 12:09:
(SendTrackingRequest)		Hidden	Shared	10/19/99 08:41:
(wAddressOpen)		Hidden	Shared	10/19/99 08:41:
(wAddressSave)		Hidden	Shared	11/02/99 04:16:
(wAppointmentOpen)		Hidden	Shared	10/19/99 08:41:
(wAppointmentSave)		Hidden	Shared	10/19/99 08:41:

Sign the SametimeQueryOpen agent

The SametimeQueryOpen agent uses the ID of the agent's signer to read the Secrets database and obtain a token. Because the Secrets database is used in the authentication scheme, access to it is strictly controlled. Therefore, the SametimeQueryOpen agent must be signed by some entity with access to the Secrets and Tokens databases. That entity can be a server's or person's Notes ID. If you do not have manager access to these databases, contact your server administrator to gain access.

Once you have the proper permissions, you can sign the agent. To do this:

1. Switch to the Notes ID.
2. Open the mail template database in Domino Designer.
3. Double-click the SametimeQueryOpen agent to open it.
4. Click File - Save to save the agent.
5. Close the Agent.

Tip To verify that the agent was successfully signed, right-click the SametimeQueryOpen agent and choose Properties. Click the Fields tab and check that the $UpdatedBy field contains your name.

Scripting the mail template

We will now add LotusScript code to the mail template to enable it to provide Who Is Online awareness. The steps to this process are as follows:

1. Initialize the Sametime script libraries.
2. Log in to the Sametime community.
3. Build a user list of people using a field on a form.
4. Create a Who Is Online action button on the form.
5. Issue OpenWhoIsOnlineWindow call.
6. Issue CloseWhoIsOnlineWindow call.

Initialize the Sametime script libraries

We will now initialize the Sametime script libraries copied to our application. The best place to do this is from the Database Script in the mail template's Memo form.

Replace the code in the *Options* event of the (Database Script) with this code:

```
Option Explicit

Use “VP”

Use “SametimeClient”
```

This adds the “VP” and “SametimeClient” scripts from the script library and initializes the Virtual Places (VP) functionality. Virtual Places is a patented technology that allows users to be aware of which other users are in an online environment and working in the same “virtual place”. This technology was developed by Ubique, a subsidiary of Lotus/IBM.

Log in to the Sametime community

The *vpLogin* command logs users in to the Sametime server after checking the LSX version and retrieving the Sametime server and user name from the Notes database. Multiple vpLogin commands may be issued from different databases, but the user will only be logged in to the Sametime server once. There are two methods to log in to the Sametime server:

- Get a token, then log in with that token. See Chapter 2 for a discussion on Secrets and Tokens.
- Log in with a user name and password.

The second option, logging in with a user name and password, forces users to actively log in to the Sametime server every time they enter a Sametime-enabled database. This repetitious action can become annoying over time.

We will log in using the token method instead. To log in using the token method, add this code to the *Postopen* event of the (Database Script):

```
Sub Postopen(Source AsNotesuidatabase)
   ’ check if logged in already
   If (vp.isLoggedIn) Then
     ’ do Nothing
   Else
     Dim token as String
     Token = SametimeProfileGetToken()
     Call vpLogin(token)
   End If
End Sub
```

After checking to see that the user is not already logged in to the Sametime server, the login process begins. The function SametimeProfileGetToken issues a token that is then used in the vpLogin call.

To disconnect from the Sametime server after a session, use the *Logout* method. Script the *Terminate* event of the (Database Script) as:

```
Sub Terminate
    Call vp.Logout()
End Sub
```

Enable the mail form

We must add code to prepare the mail form to receive the “Who Is Online” button. This button will contain the code that affects the functionality.

In the (Globals) Memo, in the *Options* event, replace the existing code with this code:

```
    Option Explicit
    Use “VP”
```

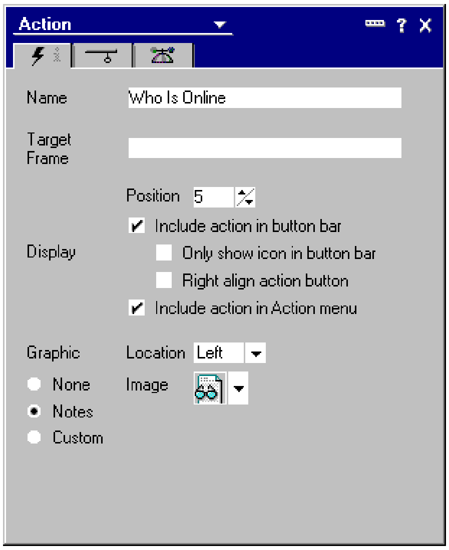
In the Memo (Form) section in the *Initialize* event, add this code to initialize the uidoc variable:

```
Sub Initialize
    Dim ws as new NotesUIWorkspace
    Dim uidoc as NotesUIDocument
    Set uidoc = ws.CurrentDocument
End Sub
```

Create a new action button on the form

In our application, the window that displays the list of online users will open when a button is clicked. This button must be created and included in the button bar of the template. To create this button:

1. Open the Memo form.
2. Choose Create - Action from the Designer Domino menu.
3. In the Name field, type Who Is Online.
4. Choose any position for the button.
5. Choose to include the action in the button bar and in the Action menu options.
6. Optionally, choose a graphic for the button.


Action

Name	Who Is Online
Target Frame	
Position	5
Display	[✓] Include action in button bar [ ] Only show icon in button bar [ ] Right align action button [✓] Include action in Action menu
Graphic	None / [● Notes] / Custom
Location	Left

This "Who Is Online" button will contain all the code necessary to:

- Open the Who Is Online window
- Display the online status of the list of users provided
- Close the Who Is Online window

Issue OpenWhoIsOnlineWindow call

The OpenWhoIsOnlineWindow UI method opens the pop-up window that displays who is online and their status based on a list of user names presented. This method and its parameters are provided below.

Method	*OpenWhoIsOnlineWindow*
Syntax	OpenWhoIsOnlineWindow (title As String, userNameList As String, listDelim As String) windowId As Long
Description	Creates a window containing the "Who Is Online" list
Parameters	*title* - the title of the "Who Is Online" window *userNameList* - the list of user names of the people on the "Who Is Online" list *listDelim*- the character(s) to be used as a delimiter between individual names in the userNameList
Return Value	Integer - the ID of the created window

In order to see the Who Is Online feature in action, you must provide the Sametime server with a list of names for which you want to know the online status. This is done using a field(s) on the form. We will use the TO field in this case. The field name for our example will be "names."

Place this code in the action button's *Click* event to provide the user name list and to open the "Who Is Online" pop-up window:

```
Sub Click (Source as Button)

   Dim userNameList as String

   Dim userName as NotesName

   Forall name in uidoc.FieldGetText("names")

     Set userName = newNotesName(name)

     If (userNameList = "") Then

          UserNameList = userName.Abbreviated

     Else

          UserNameList = userNameList & "," & _
   userName.Abbreviated

     End If

   End Forall

   WindowId = vp.OpenWhoIsOnlineWindow_
   ("Title",userNameList,",")
```

Call CloseWhoIsOnlineWindow to close the window

The Close WhoIsOnlineWindow UI method closes the pop-up window that displays the list of online users. This UI method and its parameters are provided below.

Method	*CloseWhoIsOnlineWindow*
Syntax	CloseWhoIsOnlineWindow(placeName as String)
Description	Closes the specified "Who Is Online" window
Parameters	*windowId* - the ID of the "Who Is Online" window to close
Return Value	None

Place this code in the action buttons *Click* event after the OpenWhoIsOnlineWindow code to close the open window:

```
   WindowID = vp.CloseWhoIsOnlineWindow

End Sub
```

Testing the template

Use the following steps to test the Sametime-enabled mail template:

1. Open the database in your Notes client.
2. Have at least two online users (User A and User B) where User A is viewing in his inbox a message from User B, and User B is online.
3. In User A's document, check that User B is online.
4. Have User A send a message to User B.

Deploying the template

Sametime-enabled applications can be deployed on a Sametime server or they can reside on a separate Domino server. In the second deployment scenario, the user would log on to the Sametime server separately to access the application. Chapter 2 provides a detailed explanation of both deployment options, as well as the advantages and disadvantages of each.

If you place Sametime-enabled applications on a Domino server, you will need to perform these specific tasks to make the applications work properly:

- Set up a connection record to schedule replication of the Secrets and Tokens databases from the Sametime server to the Domino server that will house the application.
- Ensure that the signer of the database or its agents has the authority to run the command "Run Unrestricted LotusScript/Java Agents" on the Domino server.
- Make sure that the ID used has access to the Secrets and Tokens database on the Domino server. By default, only the server IDs and the default Sametime development signature "Sametime Development/ Lotus Notes Companion Products" have access to these databases.
- Make sure that the NAB on the Domino server contains the "Sametime Server" field in the Person document and that the name of the Sametime server for your network is in the field.

You can find detailed instructions on how to perform these procedures in the Sametime Administrator's Guide. Once these tasks have been completed, Sametime-enabled applications can be placed on the Domino server.

Adding Who Is Online awareness to a Notes application using Java

We will now demonstrate how to add Who Is Online awareness to a Web application using Java applets. In Sametime-enabled Web applications, users can see each other whether they are in the same application or a replica of the application. Users of Web applications can connect to the application using a Web browser, the Connect client or a Notes client. However, unlike Notes applications, users do not have to install the Connect client to access Web applications.

Before stepping through our example, we will first introduce the Community Services API classes. These are grouped into Java packages.

The Java packages

The Java packages that contain the Community API are part of the VpAPI.jar and VpAPI.cab archive files located in the Sametime Toolkit Utilities database. These five Java packages and their classes are as follows:

Package	*Class(es)*
com.ubique.vp	VpSession.java
com.ubique.vp.services	CommunityService.java CommunityServiceListener.java InstantMessagingService.java InstantMessagingServiceListener.java WhoIsHereService.java WhoIsHereServiceListener.java WhoIsOnlineService.java WhoIsOnlineServiceListener.java
com.ubique.vp.ui	ChatUI.java ChatUIListener.java WhoIsHereUI.java WhoIsHereUIListener.java WhoIsOnlineUI.java WhoIsOnlineListener.java
com.ubique.vp.constants	Define the constants used by the Community API.
com.ubique.vp.types	Define the data types used by the Community API.

Using the Community API classes

When using the Community API semantic and UI components, there are several guidelines that you must adhere to. First, you must always create an instance of the CommunityService class. This class provides the login, name resolution, logout and other functions necessary to communicate with the Sametime server.

Second, when you add the semantic or UI component that your class implements, you must:

- Create an instance for each component
- Register the class as an event listener to each component. Each semantic and UI component has an *addListener* and *removeListener* method to register and unregister the class as a listener.
- Implement the appropriate ServiceListener.

Our example

Developers can use the Java classes provided with Sametime to build their own applets. For our purposes, however, we will use the LiveNamesApplet available on the Sametime server to demonstrate how to add Who Is Online awareness to a mail template.

When complete, the Memo form of the mail template will be able to display the online status of a list of users in a pop-up window. It will also display a message dialog box when the name of an online user is selected and double-clicked.

Preparing the template

The Sametime script libraries and agents found in the Sametime Toolkit Utilities database are necessary to enable the mail template that we will use. Therefore, these elements must be copied to the template and signed.

Add the script libraries

Copy the following Sametime script libraries from the Sametime Toolkit Utilities database (STToolkitUtils.nsf) in the *Script Libraries* view and paste them into the same view of the mail template.

Script Name	*Script Function*
SametimeAgent	A script program used in the Sametime server login process. (The source code is removed from this script.)
SametimeClient	A script program used in the Sametime server login process. (The source code is removed from this script.)
SametimeHost	Scripts used in locating a person's Sametime server.
SametimeServerName	A script program used to locate the Sametime server associated with the application.
SametimeStrings	A set of translatable strings called by the Sametime script libraries.

Add the Sametime agents

The *SametimeWebQueryOpen* and *SametimeGetHost* agents must be added to the template. The SametimeWebQueryOpen agent is used to read the Secrets database and handle authentication for users accessing the Sametime-enabled applications with a browser.

The SametimeGetHost agent locates the Sametime server for user login. During the login process, this agent first looks for the name of a user's Sametime server, then goes to the Sametime server's Domino directory. When it finds the user's Person record it reads the Sametime server field of that record. The user is then logged in to the Sametime server with that entry.

Copy the SametimeGetHost and SametimeWebQueryOpen agents from the Sametime Toolkit Utilities database in the *Agents* view, and paste them into the same view in the mail template.

Sign the agents

In order to sign the SametimeWebQueryOpen and SametimeGetHost agents, you must have manager access to the Secrets and Tokens databases. The SametimeWebQueryOpen agent must be signed with the server ID of the server on which the application will be deployed. Optionally, it can be signed by someone who has the authority to run unrestricted agents on the server on which the application will run.

The SametimeGetHost agent must be signed by someone with the application server's administrator ID or by someone with the authority to run unrestricted LotusScript Agents on the server.

Scripting the mail template

We will now add Java elements to the Memo form to enable it to provide Who Is Online awareness. There are three steps involved:

1. Add necessary fields to the Memo form.
2. Add code to the existing events.
3. Add the code elements to the Memo form.

Add fields to the forms

These two fields must be added to the form in order for the Who Is Online feature to work:

Field Name	*Purpose*
User_Name	Contains the hierarchical name of the user. This field should be a hidden computed-for-display text field with the formula " @UserName".
SAMETIME_TOKEN	This field stores the user's authentication token. The field should be a computed-for-display field with an empty string as the value formula. The actual value will by computed by the SametimeWebQueryOpen agent.

These fields should be added at the top of the form.

Add code to the WebQueryOpen event

Insert these two lines into the Memo (Form) *WebQueryOpen* event:

```
@Command([ToolsRunMacro]; "SametimeWebQueryOpen");
@Command([ToolsRunMacro]; "SametimeGetHost")
```

This code will handle the login to the Sametime server.

Add Java elements to the form

We will use the LiveNamesApplet in our example. This applet must be added to your Sametime server under the http root directory.

The LiveNamesApplet contains the VpAPI.jar, the LiveNames.html and these classes:

- LiveNamesApplet — the main class.
- LiveNamesItem — represents an item in the list.
- LiveNamesView — manages the data and display of the names list provided to the applet.

- MyVpClient — encapsulates all Sametime functions and is responsible for the connection with Sametime. It implements the WhoIsOnlineServiceListener and CommunityServiceListener.

We will use passthru HTML to add the applet to the form. For the Codebase property use:

```
codebase http://[SametimeHost]/st/
```

where [SametimeHost] is a computed-for-display text field with an empty string as the value formula. The value for this field will be computed by the "SametimeGetHost" agent.

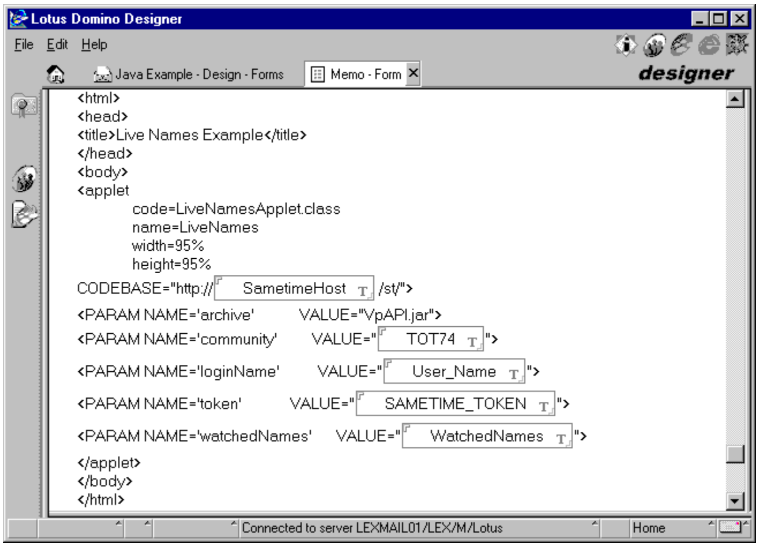
Since we will use token-based authentication to access the application, the parameters loginName and Sametime_token must be added to the form. Use the following format to do this.

- <PARAM NAME="loginName" VALUE="[USER_NAME]">
- <PARAM NAME="token" VALUE="[SAMETIME_TOKEN]">

Other parameters that will be passed to the applet are:

- Community - the Sametime server name
- WatchedNames - the user name list for which to check online status

When all the Java elements and parameters are added to the form using passthru HTML, it should look similar to this:


Lotus Domino Designer
File Edit Help
Java Example - Design - Forms
Memo - Form
designer
Connected to server LEXMAIL01/LEX/M/Lotus
Home

```
<html>
<head>
<title>Live Names Example</title>
</head>
<body>
<applet
        code=LiveNamesApplet.class
        name=LiveNames
        width=95%
        height=95%
CODEBASE="http://SametimeHost/st/">
<PARAM NAME='archive'      VALUE="VpAPI.jar">
<PARAM NAME='community'    VALUE="TOT74">
<PARAM NAME='loginName'    VALUE="User_Name">
<PARAM NAME='token'        VALUE="SAMETIME_TOKEN">
<PARAM NAME='watchedNames' VALUE="WatchedNames">
</applet>
</body>
</html>
```

The LiveNamesApplet source code

The complete class source files are available in the LiveNamesApplet sample. The LiveNamesApplet class and the MyVpClient class largely demonstrate how the Java API is used. After receiving the necessary parameters, the LiveNamesApplet class initializes the MyVpClient class. The MyVpClient class contains the Sametime API.

Let's look at some of the code behind the LiveNamesApplet class and MyVpClient, the API class. The LiveNamesApplet class shows the *init, start, resolvenames, userStatusChanged* and *create1On1Chat* methods in action. We will display the code for the init method, which handles these actions:

- Initalize the Sametime API Class
- Get the list of names to watch
- Create the list of LiveNamesItems
- Create the LiveNamesView

This is the code used in the LiveNamesApplet class for the init() method:

```
public void init()
{
     m_vpClient = new MyVpClient(this);
// Get the list of names to watch
     String watchedString = getParameter("watchedNames");
     if (watchedString != null &&_
!watchedString.equalsIgnoreCase("")) {
// Break the names string into a list of names
     m_watchedPeople = breakNameList(watchedString);
// Create the list of LiveNamesItems
     m_items = new_ LiveNamesItem[m_watchedPeople.length];
     for (int i = 0; i < m_items.length; i++)
 m_items[i] = new LiveNamesItem(m_watchedPeople[i], "", i,
      VpkUserStatus.VPK_USER_STATUS_OFFLINE);
// Create the LiveNamesView
     Scrollbar scroll = new_ Scrollbar(Scrollbar.VERTICAL);
     scroll.setBackground(Color.lightGray);
//scroll.hide();
     setLayout(new BorderLayout());
     add(“Center”, m_view = new_ LiveNamesView(m_items,
scroll, this));

     add(“East”, scroll);

 }

     setBackground(Color.white);

}
```

The MyVpClient API Class handles these actions:

- Create the Sametime API objects needed
- Create the Who Is Online watch list
- Get status change events
- Handle the resolve process

This is the code used to resolve user names:

```
public void resolveNames(LiveNamesItem[] items)

     {

 ’ Create a userNames array.

 String[] userNames = new String[items.length];

 for (int i =0; i<items.length; i++)

 userNames[i] = items[i].getUserName();

 ’ Issue a resolve request to the community.

 Integer resolveID = m_community.resolveNames(userNames, true,
false, true,false);

 ’ Keep the items.

 m_resolveItems.put(resolveID, items);

 }
```

This code creates the Who Is Online watch list after the names supplied are resolved:

```
public void namesResolved(Integer resolveID, VpResolveMatches[]
users)

 {

 Object o = m_resolveItems.remove(resolveID);

 ’ Check if it’s our request.

 if (o == null)
 return;
 LiveNamesItem[] items = (LiveNamesItem[])o;
 ’ Get the responses one by one.
 VpResolveInfo[] matches;
 m_userIds = new VpkId[items.length];
 for (int i = 0; i<users.length; i++)
 {
 matches = users[i].getMatches();
 ’ Set the corresponding item. (We take the first match only).
 items[i].setStatus(null,
VpkUserStatus.VPK_USER_STATUS_OFFLINE, “”);
 items[i].setUserId(matches[0].getId().getId());
 items[i].setUserName(matches[0].getName());
 m_userIds[i] = matches[0].getId();
 }
 ’ Pass the list of resolved users to the applet
m_applet.resolveFinished(items);
 ’ Create a Who Is Online list.
           m_wioListId = m_wio.createList();
 }
```

Test the template

Test the template’s awareness and chat capabilities by having several users log on to the Sametime community. If they can see each other online and exchange messages, then the template has been successfully Sametime-enabled with Who Is Online awareness.

Distribute the template

Once the template has been successfully tested, it can be deployed on a dedicated Sametime server or on a Domino server that incudes Sametime.

Important The preparation procedure should be repeated for each new server onto which the template is installed.

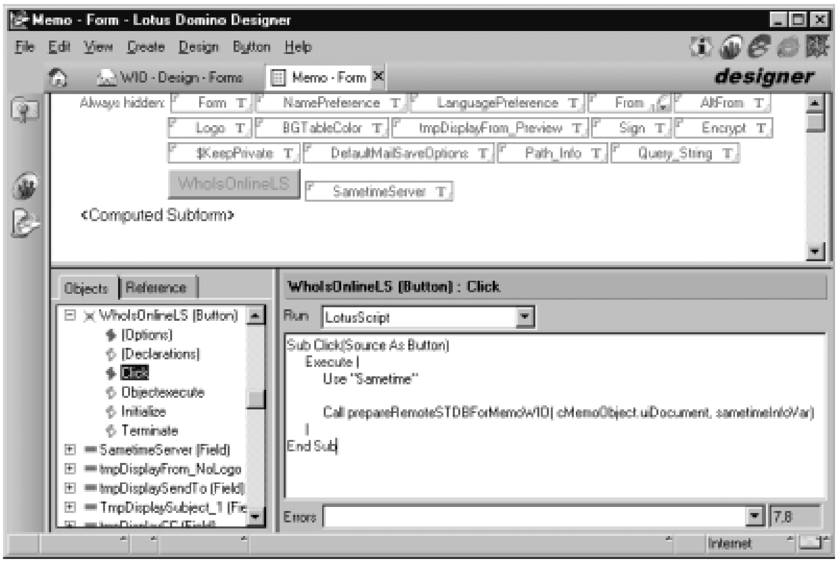
Who Is Online functionality integrated into Notes R5.0.2 mail

In Notes 5.0.2, the Sametime Who Is Online functionality has been integrated into the Notes mail template. Notes mail users will see this functionality as a clickable button on the action bars of the Memo, Reply and Reply with History forms.

If a customer has purchased Sametime and it is installed and configured within their Notes domain, the Who Is Online button will automatically appear in the mail form's action bar. This button has a hide/when formula associated with it that is based on the Sametime Home Server field in the Person record of the Domino Directory, and the installation of the Sametime Connect client. If the user has an entry in his Sametime Home Server field and has installed the Connect client, the Who Is Online action button will be displayed. The button will be hidden if either of these is not true. The Who is Online functionality is enabled in the read and/or edit mode.

The WhoIsOnlineLS field

Specifically, a WhoIsOnlineLS button has been added to each of the Memo, Reply and Reply with History forms. The diagram below shows the position of this button in a subform on the Memo form. The code associated with the *Click* event of the button is also shown.



WhoIsOnlineLS (Button) : Click

Run: LotusScript

```
Sub Click(Source As Button)
    Execute |
        Use "Sametime"

        Call prepareRemoteSTDBForMemoWID( cMemoObject.uiDocument, sametimeInfoVar)
    |
End Sub
```

Errors: 7,8

How it works

When you press the Who Is Online button within a mail message, a window opens. This window contains a Java applet which displays the online status of individuals and groups listed in the From, To, and CC fields of the form. The applet allows users to initiate a chat session with one or all of the displayed online members. Unlike Notes mail messages, an instant message appears on a person's computer screen on top of other open windows. Therefore, the receiver becomes aware of the message almost immediately.

This Who Is Online window is disconnected from the document from which it was launched. If you close the document, the window will remain open until either the user closes it or the Notes client is closed.

Copying the Who Is Online button and embedding it into another application

Developers can copy the design of the Who Is Online button from the Notes 5.0.2 mail template and incorporate it into any application. Copying the button design and embedding it into other applications involves a few steps. If you house the new Sametime-enabled application on a Domino server, it is not necessary to replicate the Secrets and Tokens databases to that server. If the application is placed on a Sametime server, however, the server must have the Secrets and Tokens databases installed.

Assumptions for this procedure

The following assumptions are made for this procedure if the application is to be accessed by the Notes client and Web browsers:

- Sametime Connect client is installed. This will ensure that the LSX API and other required components are installed.
- Sametime server field in the location document is not empty. It should contain code similar to the following:

  ```
  SameTimeServer:=@LocationGetInfo([SametimeServer]);

  @If@IsError(SameTimeServer);“”;@Name([CN];SameTimeServer))
  ```
- Sametime server has the STDomino.nsf that contains the STWIO form, the STWIO applet and the SametimeQueryOpen agent.
- Sametime server has the STAuthT.nsf Token database installed.
- Sametime server has the STAuthS.nsf Ssecrets database installed.

Copy the necessary design elements

In order for the Who Is Online button to function properly, there are several design elements that must be copied from the 5.0.2 mail template and pasted into your application. As you go through the following steps, these objects should be copied from the mail template and pasted in their respective places in the application.

- *Script Library* — Sametime. This script library contains the subroutine prepareRemoteSTDBForMemoWIO which will be modified later.
- *Shared Action* — Who Is Online. This action contains the code that opens the Who Is Online window and sets the parameters for this window.
- *Image* — act_whoisonline.gif.
- *Form design elements* - copy the WhoIsOnlineLS button and the SametimeServer field.

Make coding modifications

Once the design elements are copied to the application, the following coding modifications must be made to the form design and the Sametime script library.

1. Make these code modifications in the Form design of your application:
   - In the (Globals) *Declarations* event, add:

   ```
   Dim ws As NotesUIWorkspace
   ```

   - In the (Globals) *Initialize* event, add:

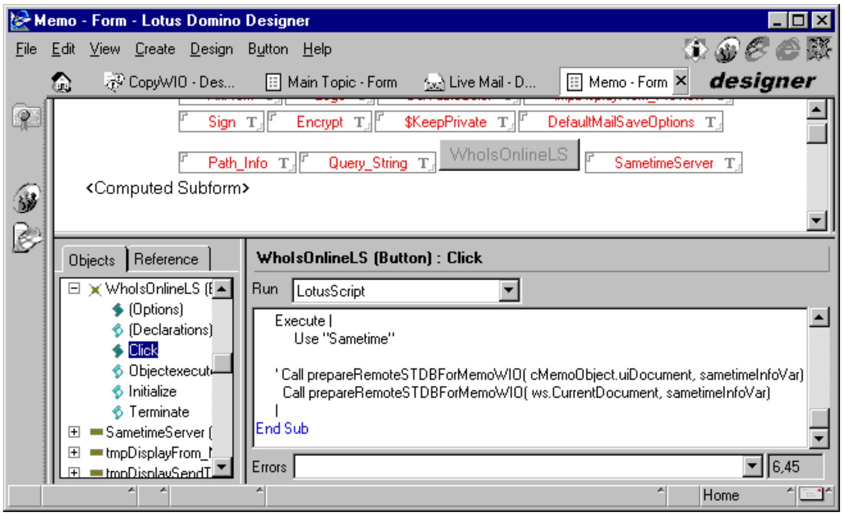
   ```
   Set ws=NewNotesUIWorkspace
   ```

2. In the Form design, insert the “Who Is Online” Shared action.
3. In the WhoIsOnlineLS button’s *Click* event, comment out this line of code:

   ```
   ’ Call prepareRemoteSTDBForMemoWIO(cMemoObject.uiDocument,
   sametimeInfoVar)
   ```

   and enter this:

   ```
   CallprepareRemoteSTDBForMemoWIO(ws.CurrentDocument,
   sametimeInfoVar)
   ```



```
Execute |
   Use "Sametime"

' Call prepareRemoteSTDBForMemoWIO( cMemoObject.uiDocument, sametimeInfoVar)
 Call prepareRemoteSTDBForMemoWIO( ws.CurrentDocument, sametimeInfoVar)
 |
End Sub
```

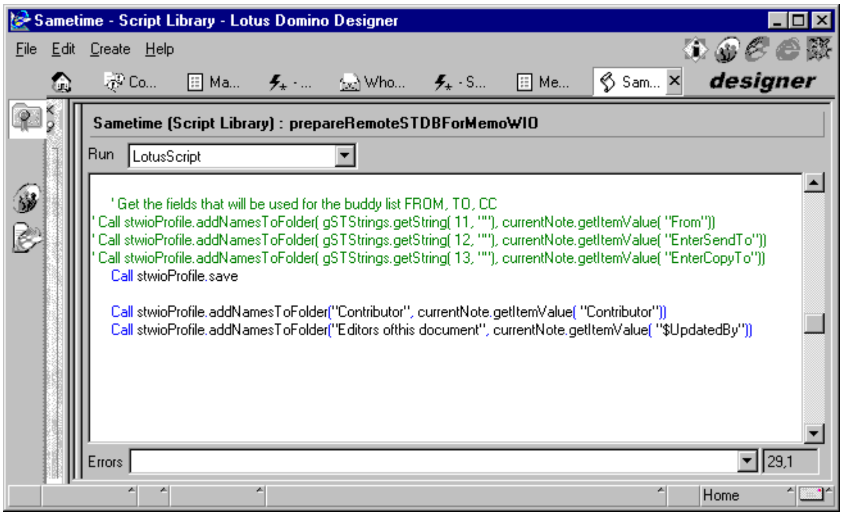
1. In the Sametime (Script Library), in the prepareRemoteSTDBForMemoWIO subroutine:

   - *Comment out* the lines of code that reference the From, EnterSendTo, and EnterCopyTo fields.

```
'Call
stwioProfile.addNamesToFolder(gSTStrings.getString(11,""),
currentNote.getItemValue("From"))

'Call
stwioProfile.addNamesToFolder(gSTStrings.getString(12,""),
currentNote.getItemValue("EnterSendTo"))

'Call
stwioProfile.addNamesToFolder(gSTStrings.getString(13,""),
currentNote.getItemValue("EnterCopyTo"))
```

   - *Replace* the commented lines with lines of code that reference the fields which will be used for the "online names" list from your form. Below is an example:

```
Call stwioProfile.addNamesToFolder("Contributor",
currentNote.getItemValue("Contributor"))

CallstwioProfile.addNamesToFolder("Editors of this
document", currentNote.getItemValue("$UpdatedBy"))
```



Sametime - Script Library - Lotus Domino Designer
File Edit Create Help
Co... Ma... - ... Who... - S... Me... Sam...
designer
Sametime (Script Library) : prepareRemoteSTDBForMemoWIO
Run
LotusScript
Errors
29,1
Home

```
    ' Get the fields that will be used for the buddy list FROM, TO, CC
' Call stwioProfile.addNamesToFolder( gSTStrings.getString( 11, ""), currentNote.getItemValue( "From"))
' Call stwioProfile.addNamesToFolder( gSTStrings.getString( 12, ""), currentNote.getItemValue( "EnterSendTo"))
' Call stwioProfile.addNamesToFolder( gSTStrings.getString( 13, ""), currentNote.getItemValue( "EnterCopyTo"))
    Call stwioProfile.save

    Call stwioProfile.addNamesToFolder("Contributor", currentNote.getItemValue( "Contributor"))
    Call stwioProfile.addNamesToFolder("Editors ofthis document", currentNote.getItemValue( "$UpdatedBy"))
```

Finishing up

After you make all these changes to your new application, you can open the application, see the Who Is Online button and use the functionality.

Chapter 5
Adding Who Is Here awareness

Sametime-enabling an application with Who is Here awareness will provide users of the application with a list of other users who are in the same place in the Sametime community. When you open a Sametime-enabled application, you are logged in to the Sametime community. You can then enter a place within the community and search for others who are also in the same place.

When you provide a place for which you want to determine online presence, the Sametime server displays a dynamic view of those individuals who are currently in that same place. The server also notifies the client when there is a change in the online status of the individuals in the place and when individuals leave. Individuals in a Sametime community fall into one of these online status categories: “I am Online,” “I am Away” or “Do not Disturb.”

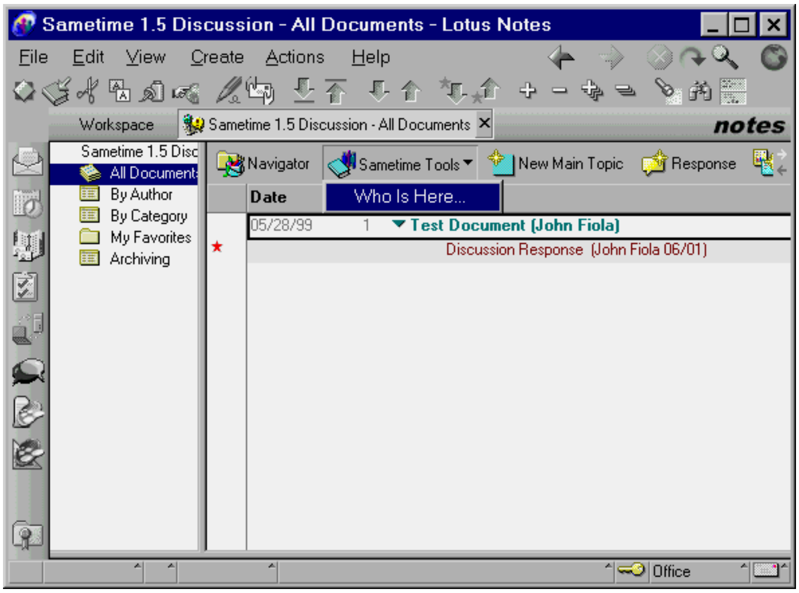
You can add Who Is Here awareness to a Notes or Domino application. Adding this awareness to a Domino application makes extensive use of standard Web programming languages such as JavaScript, HTML, and Java.

Note When users log in to the Sametime community, they are online and others can find them. This must be done first before a user can enter a place. (Who is Online awareness was discussed in Chapter 4.) A user can be online without being in a particular place, but conversely a user cannot be in a place without being online.

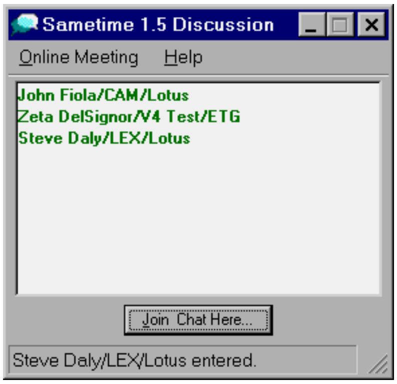
Throughout this chapter we will use two different examples of discussion databases to illustrate different ways to add Who is Here functionality to a database. We will use the Sametime enabled discussion database template that comes with the Sametime Release 1.5 product, and we will also use a Domino Release 5 discussion template that is Sametime-enabled.

Who Is Here example

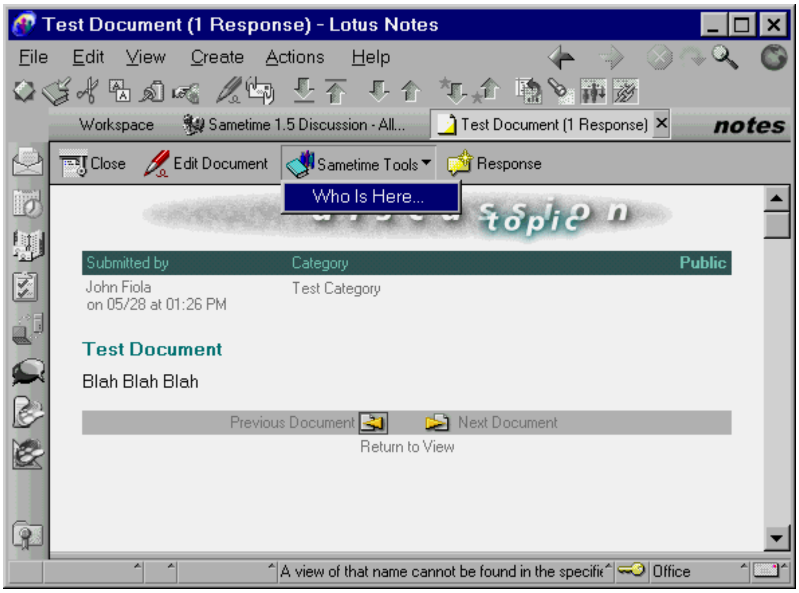
The following example demonstrates Who Is Here awareness functionality added to a discussion template. A button labeled Who Is Here has been added to the template toolbar.



When the clicks the button, a window opens showing the list of people in the discussion database and their online status. Double-click the name of an online reader to open a chat window.



Next, open a document; there is another Who is Here button.



John Fiola
on 05/28 at 01:26 PM

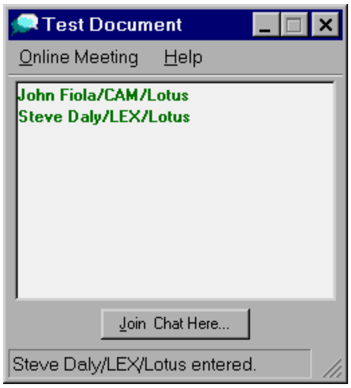
Test Category

Test Document

Blah Blah Blah

A view of that name cannot be found in the specific

This button shows who is reading the document. When you click this button, you can see this list is shorter.



John Fiola/CAM/Lotus
Steve Daly/LEX/Lotus

Steve Daly/LEX/Lotus entered.

These are the people who are reading this particular document.

Adding Who Is Here awareness to a Notes application

To demonstrate how to add Who Is Here awareness functionality to a Notes application, we will Sametime-enable a Notes discussion database. When we are finished, you will be able to access this database, find out who else is there, and optionally start a chat with them.

When the Sametime server is installed, there is a Sametime-enabled discussion template with Who Is Here functionality that is also installed on the server. We will step through a non-Sametime-enabled database to illustrate how to add basic Who Is Here functionality to a Notes application.

Setting up the environment

There are two basic scenarios under which a Sametime-enabled application can be deployed. It can be deployed directly on a Sametime server or on a separate Domino server. In the second case, the user separately logs in to the Sametime server. For a more detailed explanation of the advantages and disadvantages of each deployment method, see Chapter 2.

When the Sametime server is installed, the toolkit utilities and documentation are also installed. Included in this is a Sametime Toolkit database (STToolkitUtils.nsf) which contains all the agents and scripts that are needed to Sametime-enable a database. This database can be downloaded from the Sametime server's Web page.

Agents

Copy the SametimeQueryOpen agent from the Sametime Toolkit database into the application database. Make sure that it is signed by some entity (either by a server's or person's Notes ID) with access to the Secrets database. This agent is used to read the Secrets database in order to generate a token and uses the ID of the agent's signer. Since access to the Secrets database must be restricted, the source code for the agent has been removed.

The agent can be signed by doing the following:

1. Switch to the Notes ID to sign the agent.
2. Open the Application database in Domino Designer.
3. Open the SametimeQueryOpen Agent in Domino Designer.
4. Click File - Save.
5. Close the Agent.

Script libraries

A script library is a place where you can store reusable code segments, from other scriptable objects, that you want to use. You can code options, declarations, an initialize subroutine, a terminate subroutine, and user scripts.

Copy the following script libraries from the Sametime Toolkit Utilities database into the application database:

Script Name	*Script Function*
VP	Provides all the Virtual Places functionality. Loads the ActiveX components and handles login and logout to/from the Sametime Server.
VPStrings	A set of translatable strings called by the VP Script.
SametimeAgent	A script program used in the Sametime server login process. (The source code is removed from this script.)
SametimeClient	A script program used in the Sametime server login process. (The source code is removed from this script.)
SametimeHost	Scripts used in locating a person's Sametime server.
SametimeServerName	A script program used to locate the Sametime Server associated with the application.
SametimeStrings	A set of translatable strings called by the Sametime script libraries.

New Sametime events

Next we will put *login* and *place identification* code into the Database Scripts section when opening the database, and the *logout* and *leaveplace* code when leaving.

Note Sametime components use the concepts of *methods* and *events* to communicate with the Sametime server. Methods are the traditional function calls made to the Sametime server, and Events are calls made back to the application from the Sametime server. The *On Event* feature of LotusScript is used when looking for different events, and coding subroutines are created which are run based on trapped events. For most Java programmers, this is a common concept, but for most LotusScript programmers in Notes, this is relatively new. We illustrate this concept by showing how we program the *Log in* method and the *isloggedin* event; and also the *Logout* method and the *isLoggedout* event.

Steps for adding Who Is Here (using LSX library)

The process of adding Who Is Here awareness to a Notes application involves several basic steps.

1. Log in to the Sametime community.
2. Enter a Place.
3. Call OpenWhoIsHereWindow with the Place name.

Logging in to the Sametime community

We will add Who is Here functionality when a user opens a database. To do that, perform the following steps:

1. Open Domino Designer.
2. Open the database.
3. Choose Other.
4. Open the Database Script.
5. Add the script described in the following sections.

Logging in to the Sametime community involves two steps.

1. First, initialize the Virtual Place functionality. Using LotusScript, you call `Use VP` to initialize the script library.

   Add this code to the Options event of the Database Script:

   ```
   Use "VP"

   Use "SametimeClient"
   ```

   This code adds the VP and SametimeClient scripts from the Script Library.

2. Next, you actually log in to the Sametime community. There are two methods to do this:
   - Get a token from the Secrets database, then log in with that token. (See the Security section in Chapter 2 for a discussion of Secrets and Tokens)
   - Log in with a user name and password.

The VP script makes use of the Secrets and Tokens databases to log in to the Sametime server. This is the preferred method, since it involves no additional user intervention for authentication with the Sametime server.

Add this code to the Postopen event of the Database Script:

```
If (vp.isLoggedIn) Then

  'Do Nothing

Else

  Dim token as String

  Token = SametimeProfileGetToken()

  Call vpLogin(token)

End If
```

First we check to see if the user has already logged in to the Sametime server. If the user is already logged in, we skip any login logic.

Note The vp is not declared with a Dim statement because it is declared as a global in the VP script.

Next, if the user has not logged in, we log them in to the Sametime server. We first get the token with the API function SametimeProfileGetToken. Once the token is received, we use the vpLogin subroutine that is loaded when we load the VP script.

Responding to Sametime Events

1. Add this code to the Initialize event of the Database Script:

   ```
   On Event OpenHelp From vp Call OpenHelp

   On Event LoggedIn From vp Call LoggedIn

   On Event LoggedOut From vp Call LoggedOut
   ```

2. Add this code to the Terminate event:

   ```
   Call vp.Logout()
   ```

This will log the user out from the Sametime server.

The three lines we added in the Initialize event are events we want to globally trap when they are returned by the Sametime server. For the three events, OpenHelp, LoggedIn and LoggedOut, we want to call three different subroutines to run when each of these events occur. Next, we need to create them.

3. Move to the bottom of the code window and type the following to create the subroutines:

   ```
   Sub OpenHelp(theLsx As VpLsx)

     Call vpOpenHelp()

   End Sub
   ```

The subroutine OpenHelp will run Sametime's Help function whenever it is called from the application.

```
Sub LoggedIn(theVp As VpLsx)

   vpIDWhoIsHere = vp.EnterPlace(vpDatabase.replicaId, _
                vpDatabase.title,0)

End Sub
```

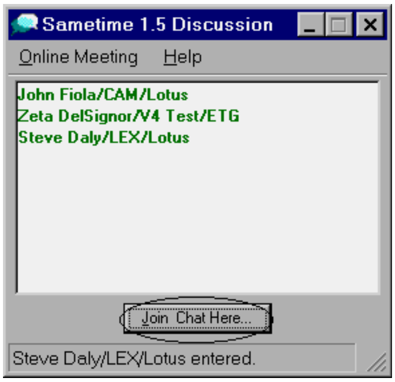
This subroutine will run when the LoggedIn event is sent from the Sametime server. When the Log-in call is made in the post open, the script will not wait for the log in to complete. The Sametime server will send back a LoggedIn Event which we trap using the On Event feature of LotusScript.

Once you have logged in, use the EnterPlace function to indicate to the Sametime server that the user is here in the database. Pass the function the database's replica ID as the name of the place; this is done as a matter of convenience. You can give it any string constant, such as HelpDesk, in the first parameter but this could lead to potential conflicts with other databases. You can use a string such as HelpDesk if you want the users to converse with people in multiple databases that share the same place name.

```
Sub LoggedOut(theVp As VpLsx,reason As Long)

   Call vp.LeavePlace(vpDatabase.replicaId)

   Call vp.LeavePlace(vpDatabase.replicaId+"chat")

End Sub
```

This subroutine will run when the LoggedOut event is trapped. When a Log out call is made in the Terminate routine, the script will not wait for the log in to complete. The Sametime server will send back a LoggedOut Event which we trap using the On Event feature of LotusScript. Once the user has logged in, we use the LeavePlace function to indicate to the Sametime server that the user is no longer here in the database.

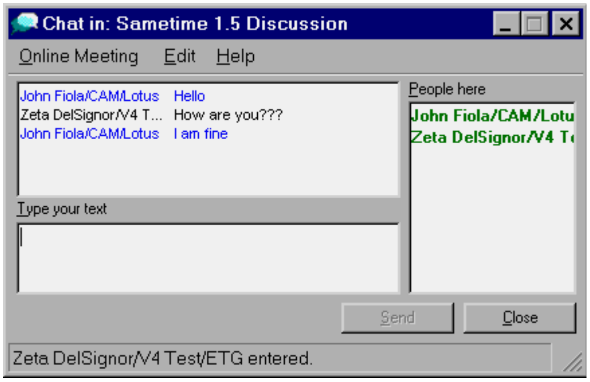
When we open the Who is Here window to see who is in the place, there is a button in the box called Join Chat Here.



John Fiola/CAM/Lotus
Zeta DelSignor/V4 Test/ETG
Steve Daly/LEX/Lotus

Steve Daly/LEX/Lotus entered.

When you click the button you enter a place with the same string as the original place with "chat" appended. We add the second LeavePlace call to account for leaving this place when the chat is closed. The following figure shows the Chat dialog.



John Fiola/CAM/Lotus Hello
Zeta DelSignor/V4 T... How are you???
John Fiola/CAM/Lotus I am fine

John Fiola/CAM/Lotu
Zeta DelSignor/V4 T

Zeta DelSignor/V4 Test/ETG entered.

Displaying Who Is Here

Once all this code is added to the Database Script section, the database is Sametime-enabled.Users can open a database and announce to the community that they are online and they are also in the database. However, this does not automatically open a dialog box showing who else is in the database. This is done with a separate function call. The function call is the OpenWhoIsHereWindow method. We can display the Who is Here function for either the database or for a document.

Calling OpenWhoIsHereWindow for a database

1. In the default view, create an action button and label it "Who's in the db." Create a new action button. In the Designer, open the default view. Select: Create - Action.
2. Select LotusScript as the Run type and Script the Action Button.
3. Add this code to the Options section:

```
Option Explicit

Use "VP"

Use "SametimeClient"
```

4. Add this code to Declarations:

```
Dim toWait As Integer
```

5. Add this to the "Click" event:

```
Sub Click(Source As Button)

    Dim token As String

    Dim loggedIn As Integer

    loggedIn = vp.IsLoggedIn
    If (loggedIn) Then
         Call vp.OpenWhoIsHereWindow(vpDatabase.replicaId)
    Else
         If Not vpdatabase.Server = "" Then
              toWait = 1
              token = SametimeProfileGetToken()
              vpLogin(token)
         End If
    End If
End Sub
```

6. Add this to the initialize event:

```
Sub Initialize
    Set vpUIDocument = vpUIWorkspace .CurrentDocument
    toWait = -1
    On Event LoggedIn From vp Call LoggedIn
End Sub
```

7. Create a new subroutine and call it "LoggedIn":

```
Sub LoggedIn(theVp As VpLsx)
    If (toWait = 1) Then
         vpIDWhoIsHere =
vp.EnterPlace(vpDatabase.replicaId,_
         vpDatabase.title,1)
         Call
vp.DisplayWhoIsHereWindow(vpDatabase.replicaId)
         toWait = -1
    End If
End Sub
```

Adding Who is Here for a document

In this section we will show the steps necessary to add Who is Here functionality to a document. Once the document has been Sametime-enabled with this functionality, we can answer the question, "Who else is reading this document?"

The process of adding Who Is Here awareness to a Notes application involves several steps, which can be grouped into these basic steps:

- Check the Login status to the Sametime community and log in if necessary.
- Determine the document's ID and Enter a Place with it.
- Call the OpenWhoIsHereWindow function with the document ID Place.

The detailed steps are as follows:

1. First, initialize the Virtual Place functionality in the Form's Globals section. Using LotusScript, call **`Use VP`** to initialize the script library.

   Add this code to the Options event:

   ```
   Use "VP"

   Use "SametimeClient"
   ```

2. Next, in the Declarations section, add:

   ```
   Dim toWait As Integer

   Dim locationId As String

   Dim displayName As String
   ```

3. Next, check to see if you are already logged in to the Sametime community, and log in if you are not. There are two methods to do this.

   - Get a token, then log in with that token.
   - Log in with a user name and password.

   The VP script makes use of the Secrets and Tokens databases to log in to the Sametime server. This is the preferred method since it involves no additional user intervention for authentication with the Sametime server.

   Add this code to the Postopen event of the Main Form:

   ```
   Set vpUIDocument = vpUIWorkspace .CurrentDocument

   toWait = -1

   vpIDWhoIsHere = -1

   locationId = vpUIDocument.Document.UniversalID

   Dim isLogin As Integer

   isLogin = vp.IsLoggedIn

   If (isLogin) Then

      displayName = vpUIDocument.FieldGetText("Subject")
   ```

```
   If (displayName = "") Then displayName = _
        vpUIDocument.WindowTitle
   vpIDWhoIsHere = _
      vp.EnterPlace(locationId,displayName,0)
End If
On Event LoggedIn From vp Call LoggedIn
On Event LoggedOut From vp Call LoggedOut
```

We first check to see if the user is already logged in to the Sametime server. If they are already logged in, then we enter the document Place by entering the document's ID as a parameter in the EnterPlace function.

If the user is not logged in to the Sametime server for some reason (for example, if the Sametime server was brought down and brought back up between the time the database was opened and the time the document was opened), we log the user in again and once again enter the database Place.

The last three lines we added in the Postopen event are events we want to globally trap when they are returned by the Sametime server. For the two events, LoggedIn and LoggedOut, we want to call two different subroutines to run when each of these events occur.

4. Next, we need to create the subroutines.

   Create the subroutines by going to the bottom of the "Initialize" section and entering the following code:

```
Sub LoggedIn(theVp As VpLsx)
  vpIDWhoIsHere = vp.EnterPlace(vpDatabase.replicaId, _
                 vpDatabase.title,0)
End Sub
```

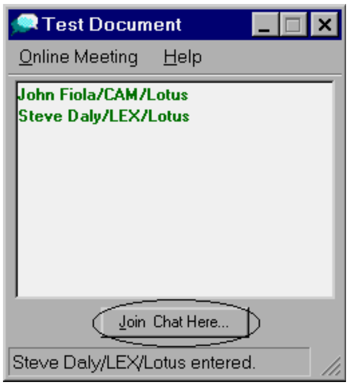
   This subroutine will run when the LoggedIn event is trapped. When the Log in call is made in the Postopen event, the script will not wait for the log in to complete. The Sametime server will send back a LoggedIn Event which we trap using the On Event feature of LotusScript. Once the user has logged in, we use the EnterPlace function to indicate to the Sametime server that the user is here in the database. We give the function the database's replica ID as the name of the place. We want to maintain the login and the place awareness at the database level, so that if the user is disconnected from the Sametime server and needs to log in again, they can do so in a simple way.

5. We then add the LoggOut subroutine as follows:

```
Sub LoggedOut(theVp As VpLsx,reason As Long)
   Call vp.LeavePlace(vpDatabase.replicaId)
   Call vp.LeavePlace(vpDatabase.replicaId+"chat")
End Sub
```

This subroutine will run when the LoggedOut event is trapped. When a Logout call is made in the Terminate routine, the script will not wait for the login to complete. The Sametime server will send back a LoggedOut Event which we trap using the On Event feature of LotusScript. Once the user has logged in, we use the LeavePlace function to indicate to the Sametime server that the user is no longer here in the database.

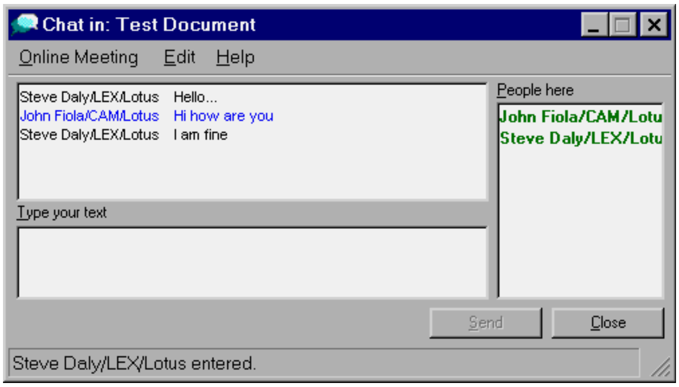
When we open the Who is Here window to see who is in the place, there is a button in the box called Join Chat Here.



John Fiola/CAM/Lotus
Steve Daly/LEX/Lotus

Steve Daly/LEX/Lotus entered.

When we click the button we enter into a place with the same string as the original place, with "chat" appended. We add the second LeavePlace call to account for leaving this place when the chat is closed. The following figure shows the Chat dialog.



Steve Daly/LEX/Lotus Hello...
John Fiola/CAM/Lotus Hi how are you
Steve Daly/LEX/Lotus I am fine

John Fiola/CAM/Lotu
Steve Daly/LEX/Lotu

Steve Daly/LEX/Lotus entered.

6. We also add some cleanup code when exiting a document. In the QueryClose event enter:

```
Call vp.LeavePlace( locationId)

Call vp.LeavePlace( locationId+"chat")
```

Who is Here dialog

Now we have a place name for every main document that is opened. When users enter the main document, they enter a place which is the document ID. However, they do not automatically see who else is reading the document. This section explains how to bring up a Who is Here dialog.

1. On the default form, create an action button and label it "Who Is Here."
2. Create a new action button by choosing Create - Action from the menu.
3. Enable the Main form so that for documents that are opened, users will enter a place where they can see who else is reading the document.
4. Script the Action button as follows:

In the Options section of the Action, we add:

```
Option Explicit
```

In the Declarations section of the Action, we add:

```
Dim toWait As Integer

Dim locationId As String

Dim displayName As String
```

In the Click section of the Action, we add:

```
Sub Click(Source As Button)

     Dim token As String
```

```
      Dim loggedIn As Integer
      loggedIn = vp.IsLoggedIn
      If (loggedIn) Then
            Call vp.OpenWhoIsHereWindow(locationId)
      Else
            If Not vpdatabase.Server = "" Then
                  toWait = 1
                  token = SametimeProfileGetToken()
                  vpLogin(token)
            End If
      End If
End Sub
```

In the Click routine, we first check to see if the user is already logged in to the Sametime server. If they are already logged in, we enter a place using the document's ID as the place name.

In the Initialize section of the Action, we add:

```
Sub Initialize
      Set vpUIDocument = vpUIWorkspace .CurrentDocument
      toWait = -1
      locationId = vpUIDocument.Document.UniversalID
      displayName = vpUIDocument.WindowTitle
      On Event LoggedIn From vp Call LoggedIn
End Sub
```

In the Initialize section we get the Document's ID to identify the name of place for the document. We also want to trap the LoggedIn event here in case the user was not logged in to the Sametime server and needed to log in to it again.

Next we create the LoggedIn subroutine.

```
Sub LoggedIn(theVp As VpLsx)
      If (toWait = 1) Then
            displayName = vpUIDocument.FieldGetText("Subject")
            If (displayName = “”) Then displayName = _
                  vpUIDocument.WindowTitle
```

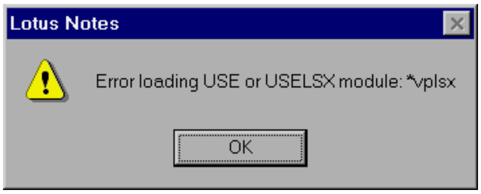
```
                vpIDWhoIsHere = vp.EnterPlace(locationId, _
                     displayName,0)
                Call vp.OpenWhoIsHereWindow(locationId)
                toWait = -1
           End If
      End Sub
```

Usability considerations

There is a usability issue when accessing the ActiveX components using the LSX. Everyone who accesses the database or the Sametime functions must have the Sametime Connect Client installed on their machine for everything to work properly. When users install the Sametime Connect Client, two dll files necessary for integration with Domino applications are installed. The files and their functions are as follows:

File Name	*Function*
vplsx.dll	Used to load the API functions. Communicates with the file vplsxapi.dll Created from the LSX toolkit
vplsxapi.dll	Used as a container for the ActiveX components. Creates instances of the ActiveX components and communicates with them through OLE

If users try to access a database when they do not have the Sametime Connect Client installed, they will get the following error:


Lotus Notes

Error loading USE or USELSX module: *vplsx

In the Options section of the VP script in the Script Library, there is the following line of code:

```
Uselsx "*vplsx"
```

This call looks on the user's local machine for the file vplsx.dll to load the Sametime components. If the Sametime Connect Client is not installed, the script will not find the file on the user's machine and, therefore, the user will receive the message shown prevviously when opening the Sametime-enabled database.

Since this call for the LSX is in the Options section, and since the LSX cannot be found, the loading error occurs. This is a Notes limitation. This error message cannot be trapped. Ideally, a developer would want to trap the message and hide the Sametime features.

Tip The workaround is to use the USELSX method instead of the UseLsx option Set up an environment variable, which indicates that the user has Sametime installed. Check for and exit any subroutine that indicates that the Connect Client has not been installed. This workaround requires the user to actively turn on the Sametime features of the database.

To do this, go to the Script Library and create a new script. In the Initialize section add:

```
Dim ses As New NotesSession
Dim ws As New NotesUiWorkspace

On Error Goto stError
If ses.GetEnvironmentString( "MyStApp" ) = "1" Then
     Call ws.UseLSX( "*VPLSX" )
End If

Exit Sub

stError:
Messagebox "Sametime Error", 64, _
           "Sametime Extensions not loaded"
Call ses.SetEnvironmentVar( "MyStApp", "0" )
Resume Next
```

Use this script in the options section of any Action button that would call the Use VP script.

Another way to avoid problems with the availability of the ActiveX components is to avoid using them entirely. Developers have the ability to create components in the form of Java applets. They can then use these applets directly in an application that is accessed by a Notes client. We describe how to do this in the next section.

Performance considerations

When logging in to a database that is Sametime-enabled, in addition to opening the application database, the user also logs in to the Sametime server. It takes additional time to perform this second login, in which the users may or may not want to activate the Sametime features. One way to avoid the login is to code in an environment variable which indicates that the user does not want to use the Sametime features. An environment variable is an entry in the Notes.ini configuration file that is set and read by Notes applications.

The code below is supplied in the templates and can be used across different applications. In the Postopen event of the Database script, it reads the environment variable and, if the user has set the Sametime awareness to be off, it exits the subroutine and prevents the user from logging on to the Sametime server.

```
Dim awareness As String

awareness = vpSession.GetEnvironmentString("st_awareness")

If (awareness = "") Then

     awareness = "1"

     Call vpSession.SetEnvironmentVar( "st_awareness","1")

End If

If (awareness ="0") Then Exit Sub
```

Steps for adding Who Is Here using Java applet components

When adding Who is Here functionality in applications accessed with a Notes client, we can add the functionality using Java Applets instead of the ActiveX components loaded using the LotusScript Extensions (LSX). The advantages to doing this are:

- Users do not have to install the Sametime Connect Client in order to use the Who is Here functionality.
- Developers have greater control of the user interface.
- The applets can be used within both Notes clients and Web browsers.

There is only one supported Who is Here applet that comes with Sametime. This is used to add Who is Here functionality to the Sametime-enabled discussion template. Since the source code is not available for the applet, we can only show how to add it to the application, but not what it does.

The Sametime server contains a set of examples for Who is Here in the toolkit. These applets have the source code available so that Java Applet developers can create and modify their own. The applets in the toolkit are considered to be samples and therefore are not supported by Lotus. It is up to the developer to build their own applets for use in their application.

Once an applet is ready for use in an application, it must be added to the Notes database. The database must first be prepared by adding agents and scripts to feed parameters to the applet.

Embedding an applet in a Notes application

Agents

Copy the SametimeQueryOpen Agent from the Sametime Toolkit database into the application database. Make sure that it is signed by some entity (either by a server's or person's Notes ID) with access to the Secrets database. This agent is used to read the Secrets database in order to generate a token, and uses the ID of the agent's signer. Since access to the Secrets database must be restricted, the source code for the agent has been removed.

The Agent can be signed by doing the following:

1. Switch to the Notes ID to sign the agent.
2. Open the Application database in Domino Designer.
3. Open the SametimeQueryOpen Agent in Domino Designer.
4. Click on File - Save.
5. Close the Agent.

Script libraries

A script library is a place where you can store code segments, from other scriptable objects, that you want to use. You can code options, declarations, an initialize subroutine, a terminate subroutine, and user scripts.

Copy the following script libraries from the Sametime Toolkit Utilities database into the application database:

Script Name	*Script Function*
SametimeAgent	A script program used in the Sametime server login process. (The source code is removed from this script.)
SametimeClient	A script program used in the Sametime server login process. (The source code is removed from this script.)
SametimeHost	Scripts used in locating a person's Sametime server.
SametimeServerName	A script program used to locate the Sametime Server associated with the application.
SametimeStrings	A set of translatable strings called by the Sametime script libraries.

Note We are not using the VP and VPStrings scripts in this instance. The VP script is responsible for accessing, loading, and manipulating the ActiveX components on the user's workstation that were installed when the user installed the Connect Client.

Adding a Who is Here applet

Fields to add on a form

When creating a new Lotus Notes form, there are several fields that are used as parameters in any Java applet that must be added in order for the Who is Here feature to work using a Notes client. The fields and their purposes are as follows:

Field Name	*Purpose*
User_Name	Identifies the name of the user
SAMETIME_TOKEN	Stores the user's token
SametimeReplicaId	The Database Replica ID used to identify the database as a "Place"
SametimeDocumentID	The Document ID used to identify a document as a "Place"
SametimeServer	Identifies the Sametime server to log in

These are the minimum fields that must be placed on the form. A developer can create a custom Java applet with Who is Here functionality, which might require fields that set additional applet parameters. These additional fields should be added at this point. The datatype for these fields should be Computed for Display to prevent the data in these fields from being stored in the document.

The recommended way to add these fields to the form is to add them to a subform. That way you can manage them all in one place. Set the Hide When properties for these fields so that they are not displayed. They should have a datatype of "Text" with a default set to null ("").

Embedding an applet

First, you need to create a form or subform in which to embed the Java applet. We recommend that you create a subform and embed the applet. This way you can add and maintain it in one place and can use it in multiple forms.

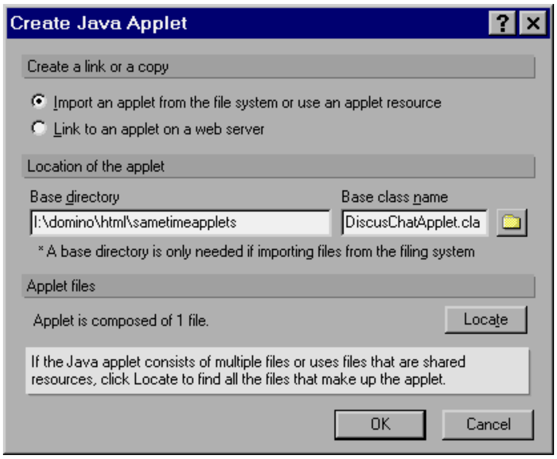
Storing applets

For any Sametime-enabled database that is accessed through a browser, a set of Java applets are used to log in to a Sametime server and start conversations. Due to the Java security model, the code must be placed directly on the Sametime server. Production environments should already have procedures in place to handle this. In Domino Release 5 we can store the applets directly inside the application database. For applications that use only the Notes client this can be a viable alternative.

Use the following steps to store a Java applet:

1. Create a form or subform where the applet will be stored.
2. Embed the applet by clicking Create - Java Applet.

   Here you can choose to load the applet from the database or from a file server.



Create Java Applet

Create a link or a copy

(●) Import an applet from the file system or use an applet resource
( ) Link to an applet on a web server

Location of the applet

Base directory: I:\domino\html\sametimeapplets
Base class name: DiscusChatApplet.cla

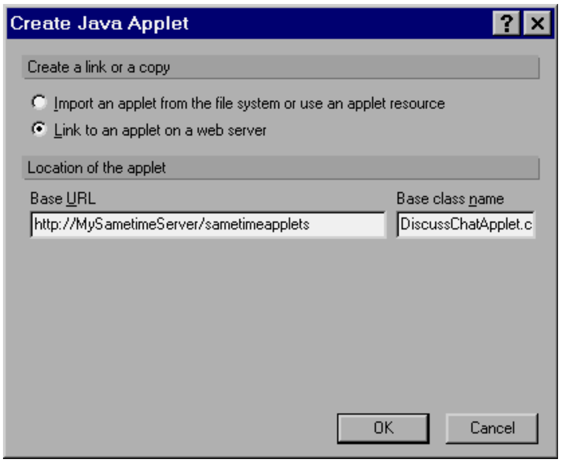
* A base directory is only needed if importing files from the filing system

Applet files

Applet is composed of 1 file.

If the Java applet consists of multiple files or uses files that are shared resources, click Locate to find all the files that make up the applet.

Alternatively, you can link to an applet on the Sametime server.



Create Java Applet

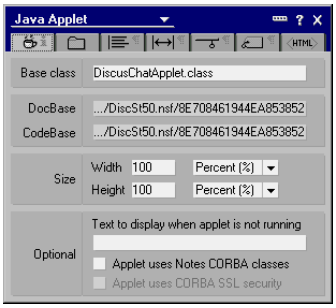
Create a link or a copy

( ) Import an applet from the file system or use an applet resource
(●) Link to an applet on a web server

Location of the applet

Base URL: http://MySametimeServer/sametimeapplets
Base class name: DiscussChatApplet.c

The Java applet's properties are shown in the following figure.



Base class: DiscusChatApplet.class

DocBase: .../DiscSt50.nsf/8E708461944EA853852

CodeBase: .../DiscSt50.nsf/8E708461944EA853852

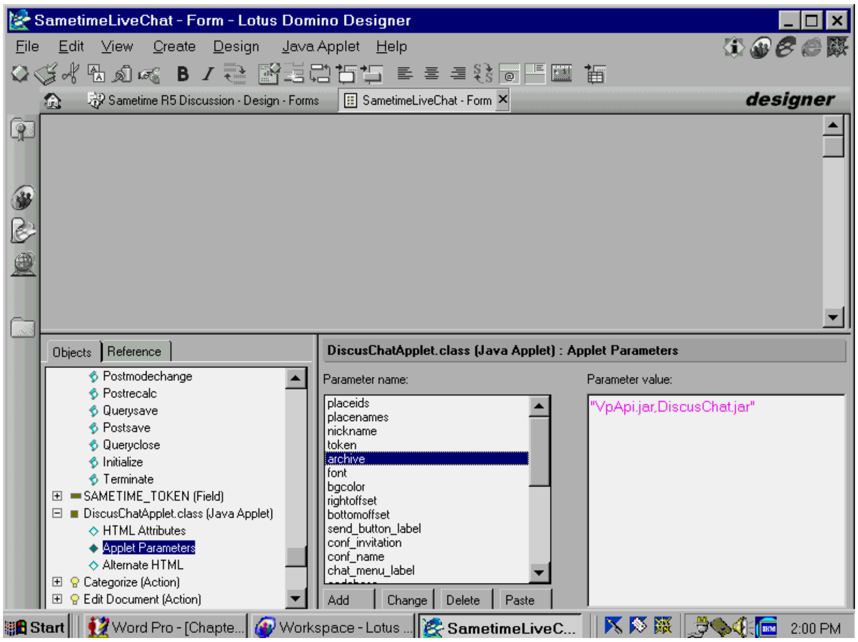
Size: Width 100 Percent (%); Height 100 Percent (%)

Optional: Text to display when applet is not running

Applet uses Notes CORBA classes

Applet uses CORBA SSL security

Add the applet's parameters. The parameters can be plain text, they can refer to any @functions such as @UserName, or they can refer to a field on the form.



Postmodechange
Postrecalc
Querysave
Postsave
Queryclose
Initialize
Terminate
SAMETIME_TOKEN (Field)
DiscusChatApplet.class (Java Applet)
HTML Attributes
Applet Parameters
Alternate HTML
Categorize (Action)
Edit Document (Action)

DiscusChatApplet.class (Java Applet) : Applet Parameters

Parameter name:

placeids
placenames
nickname
token
archive
font
bgcolor
rightoffset
bottomoffset
send_button_label
conf_invitation
conf_name
chat_menu_label

Parameter value:

"VpApi.jar,DiscusChat.jar"

3. Save the subform and call it SametimeWIHSubform.

4. Create a new form and add the subform to it. If you want to embed this into the form, we recommend that you load it as a computed subform. That way you do not embed a copy of the applet into every document. One possible computed formula for loading the applet subform is:

   ```
   @If(@IsNewDoc | @IsDocBeingEdited ; "";
   "SametimeWIHsubform")
   ```

   This would prevent the applet subform from being loaded and therefore saved in the document when the document is being created or edited.

   In the Options section of the form add:

   ```
   Use "SametimeClient"
   ```

   ```
   Use "SametimeHost"
   ```

   These will load the necessary functions to obtain the login and server information.

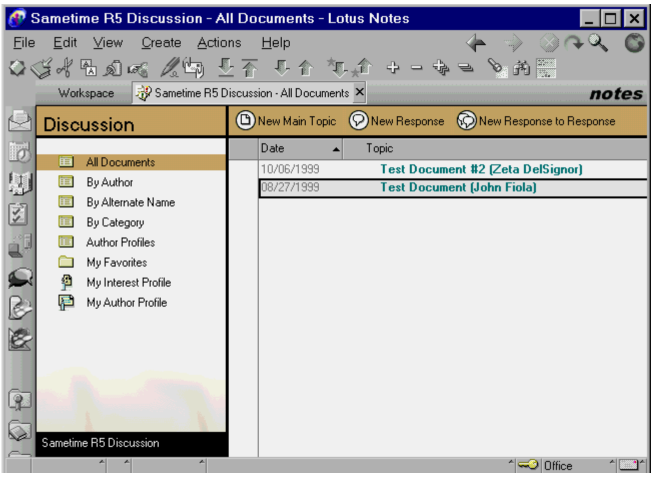
5. Finally, in the Postopen event, set any document fields that are used as parameters in the applet.

Using JavaScript to run an applet in a Notes client

Another way to run the applet is to use JavaScript to load and run it. This eliminates the possibility of embedding the applet in every document and lets you control the display of the applet.

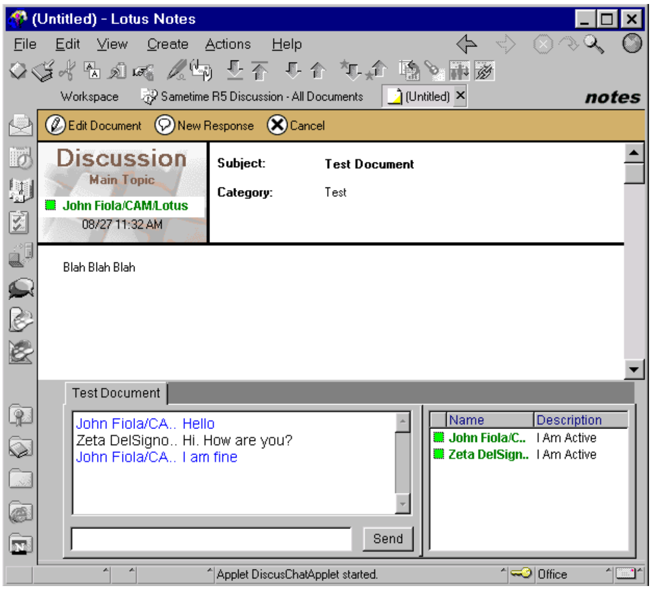
Example application (R5 Discussion Database)

The following example demonstrates Who Is Here awareness functionality added to a Domino Release 5 discussion template. The next figure shows what the user sees when they open the database.



Date	Topic
10/06/1999	Test Document #2 (Zeta DelSignor)
08/27/1999	Test Document (John Fiola)

When they click a document to open it, a frame opens at the bottom and an applet is loaded in the bottom frame showing who is in the document.



John Fiola/CAM/Lotus
08/27 11:32 AM

Subject: Test Document

Category: Test

Blah Blah Blah

John Fiola/CA.. Hello
Zeta DelSigno.. Hi. How are you?
John Fiola/CA.. I am fine

Name	Description
🟩 John Fiola/C..	I Am Active
🟩 Zeta DelSign..	I Am Active

Applet DiscusChatApplet started.

The frame in the lower right corner shows who is reading the document and their online status. The frame in the lower left shows the chat conversation that is going on among the readers of the document. The top line of this frame shows the conversation thread. The bottom is for the user to enter text for their participation in the conversation.

In the next section we show how this applet was added to the database and how it is activated.

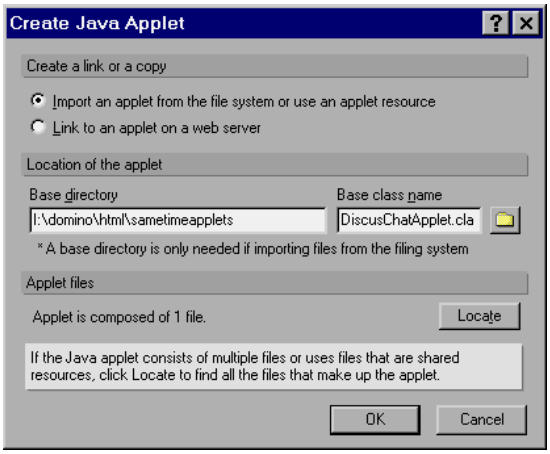
Adding Who is Here functionality using JavaScript

Once the database is Sametime-enabled, the basic steps to add the Who Is Here applet are as follows:

1. Open the database in Domino Designer.
2. Create a new form and link the applet in the form.

   Do this by clicking Create - Java Applet.

   Here you can choose to load the applet from the database or from a file server.



Create Java Applet

Create a link or a copy

(●) Import an applet from the file system or use an applet resource
( ) Link to an applet on a web server

Location of the applet

Base directory: I:\domino\html\sametimeapplets
Base class name: DiscusChatApplet.cla

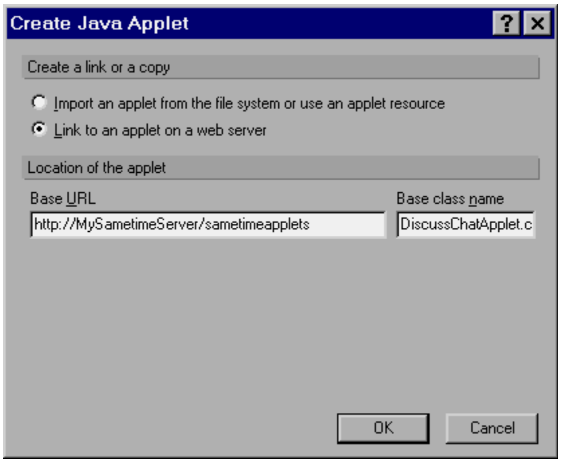
* A base directory is only needed if importing files from the filing system

Applet files

Applet is composed of 1 file.

If the Java applet consists of multiple files or uses files that are shared resources, click Locate to find all the files that make up the applet.

   Alternatively, you can link to an applet on the Sametime server.



Create Java Applet

Create a link or a copy

( ) Import an applet from the file system or use an applet resource
(●) Link to an applet on a web server

Location of the applet

Base URL: http://MySametimeServer/sametimeapplets
Base class name: DiscussChatApplet.c

3. Add the appropriate applet parameters to the applet.
4. Add a Computed for Display field called SAMETIME_TOKEN to the form.

5. In the Postopen event obtain a token and set the SAMETIME_TOKEN field to its value.

```
Dim token As String
Dim session As New NotesSession
Dim doc As NotesDocument
Dim item As NotesItem
Set doc = Source.Document
token = SametimeProfileGetToken()
Set item = doc.ReplaceItemValue("SAMETIME_TOKEN", token)
```

6. Save and close this new form.
7. Add functions to the JavaScript Header (JS Header) of the Main form from the discussion template.

```
var newFrame
```

This function sets up various parameters for later functions:

```
function authProf() {
 var pathname = window.location.pathname;
 filename =  pathname.substring(0,
        (pathname.lastIndexOf("nsf")+4))
 key = document.forms[0].AbrFrom.value


 var newWindow = window.open(filename +
         "LookupPersonalProfiles/" + key +
         "?OpenDocument&hw=1",
         "secondary_window",
         "toolbar=no,location=no,
         scrollbars=yes,
         directories=no,height=500,width=625")
}
```

The function below converts spaces in a string to +:

```
function spaceToPlus(param) {
 param = "" + param; // just to be sure this is a string
 var splittedParam = param.split(" ");
 param = "" + splittedParam[0];
 for (var i = 1; i < splittedParam.length; i++)
  param += "+" + splittedParam[i];
  return param;   }
```

The function below opens the Who is here window and runs the applet:

```
function openSametimeChatWindow() {
 // Open the WIH Form in the Bottom Frame
 var databaseName = document.forms[0].DbName.value
 var placeids = document.forms[0].SametimeDocumentID.value
 var placenames = document.forms[0].DocumentTitle.value
 var nickname = document.forms[0].loginName.value
 var token = document.forms[0].SAMETIME_TOKEN.value
 var sametimehost = document.forms[0].sametimeHost.value
 var IsWebUser = document.forms[0].isWebUser.value

 if (IsWebUser == "N")
 {
   newFrame = window.open("/" + databaseName +
        "/SametimeLiveChat?OpenForm",
        "appletFrame");
 }
 else
 {
  var urlString = "/" + databaseName +
          "/SametimeWebLiveChat?OpenForm&placeids="+
          placeids + "&placenames=" +
          placenames + "&nickname=" +
          nickname + "&token=" + token +
          "&sametimehost=" + sametimehost;
   urlString = spaceToPlus(urlString);
   newFrame = window.open(urlString, "appletFrame");
 }
}
```

Note The code above has been formated for display on the page. Pay careful attention to the line breaks.

This script does the following:

Reads in the applet parameters from the document's form.

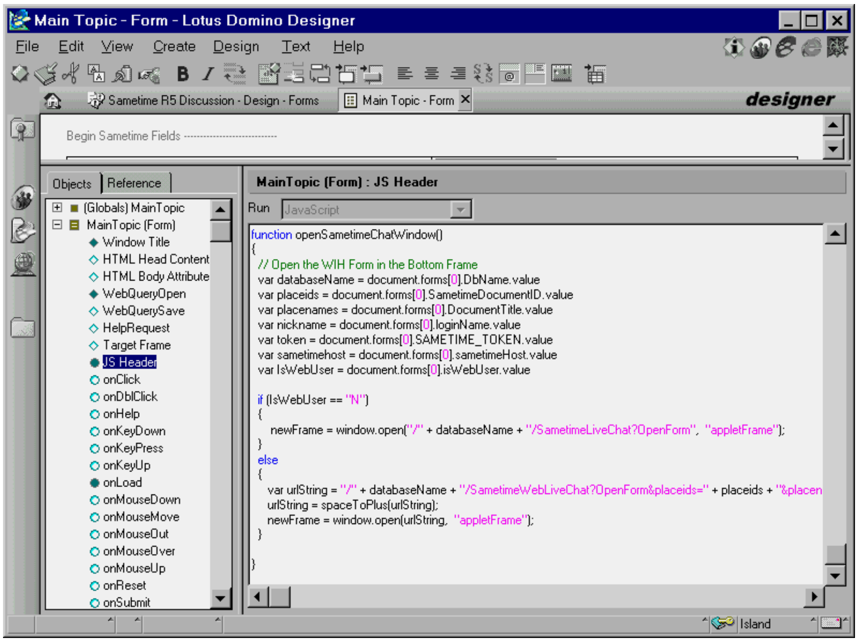
Checks the type of client the user is using, either a Notes client or Web browser.

Uses the appropriate way to load the applet. In the case of the Notes Client, it does the following:

Creates a new frame called appletFrame on the screen.

Opens the form SametimeLiveChat (this is the name of the separate form we created above) that contains the applet in the new frame.

The following figure shows how it looks in the Domino Designer.



```
function openSametimeChatWindow()
{
 // Open the WIH Form in the Bottom Frame
 var databaseName = document.forms[0].DbName.value
 var placeids = document.forms[0].SametimeDocumentID.value
 var placenames = document.forms[0].DocumentTitle.value
 var nickname = document.forms[0].loginName.value
 var token = document.forms[0].SAMETIME_TOKEN.value
 var sametimehost = document.forms[0].sametimeHost.value
 var IsWebUser = document.forms[0].isWebUser.value

 if (IsWebUser == "N")
 {
    newFrame = window.open("/" + databaseName + "/SametimeLiveChat?OpenForm",  "appletFrame");
 }
 else
 {
   var urlString = "/" + databaseName + "/SametimeWebLiveChat?OpenForm&placeids=" + placeids + "&placen
   urlString = spaceToPlus(urlString);
   newFrame = window.open(urlString,  "appletFrame");
 }

}
```

8. Call the applet in the JavaScript onLoad event.

   In the onLoad event we run the JavaScript code:

   openSametimeChatWindow();

   This will run the openSametimeChatWindow function defined in the JS Header section when the document is opened by a user.

Adding Who Is Here awareness to a Web application

The Domino server family allows us to build applications that are accessible using Web browsers as well as Notes clients. Here we describe how to add Who is Here Sametime functionality to Domino applications that are accessed strictly through Web browsers. For applications that are accessed through both Notes clients and Web browsers, you will need to integrate steps in this section with those in the previous section.

Programming tools used

For any Sametime-enabled database that is accessed through a browser, logging in to a Sametime server and starting conversations are achieved using a set of Java applets. Because of the security model of Java applets, the code must be placed directly on the Sametime server. Production environments should have procedures in place to handle this. In Domino Release 5 we can store the applets directly inside the application database. For applications that use the Notes client only, this can be a viable alternative.

The enabling of Domino applications with Who is Here functionality requires the use of Java applets to communicate with the Sametime server. Lotus does not supply an extensive list of applets that a developer can use. It is up to developers to create and customize Sametime applets for their own internal use. There is a series of sample applets, including a few Who is Here applets, for which the source code is available. They are distributed with the Sametime server as part of the toolkit and can be downloaded from the Sametime server. These sample applets can be used to determine how Who is Here functionality is coded and can be customized by developers for use in their applications.

Programming tools

You can use the following tools to develop Java applets:

- Java IDE
  Any IDE (IBM Visual Age for Java, Symantec Visual Cafe) that supports JVM 1.1.4 or higher
- Domino Designer
  For programming Notes databases and HTML/JavaScript code
- Other Web authoring tools, such as Notepad, Fusion, DreamWeaver, and FrontPage
  Used to fine-tune an application

You can create applets that do not require use of the Domino application server to run, but that is beyond the scope of this publication.

Adding Who Is Here functionality to the database

When the Sametime server is installed, the toolkit utilities and documentation are also installed. Included is a Sametime Toolkit database (STToolkitUtils.nsf) which contains all the agents and scripts that are needed to Sametime-enable a database. This database can be downloaded from the Sametime server's Web page.

Before we add specific Who Is Here functionality to the database, we need to add agents and scripts from the Sametime Toolkit database. In this section we describe the necessary agents and scripts which allow us to add the functionality.

Adding the Agents

1. Copy the SametimeWebQueryOpen Agent from the Sametime Toolkit database into the application database.

   Make sure that it is signed by some entity (either a server's or person's Notes ID) that has access to the Secrets database. This agent is used to read the Secrets database in order to generate a token, and uses the ID of the agent's signer. Since access to the Secrets database needs to be restricted, the source code for the agent has been removed.

2. Copy the following LotusScript agents from the Sametime Toolkit Utilities database:

Agent Name	*Agent Function*
(SametimeGetHost)	Identifies and sets the field indicating the Sametime server in which to log in.
(SametimeGetPlaceName)	Sets a field called SametimeReplicaId with the database's replica ID.

Unlike the SametimeWebQueryOpen agent, the two agents listed above have their source code available. The data in the fields SametimeServer, and SametimeReplicaId set in the agents above are needed as parameters in any Java applet necessary to enable Who is Here functionality. The developer can modify how these fields are set, or they can change the names of the fields that these scripts set to meet the requirements of the application.

There may be other fields necessary to round out an application. These could be database or document parameters that are used as Java applet parameters. These additional parameters can be set by new agents created by the developer. These additional agents can also be created and added to the agent list. When creating additional agents, add in the Options section:

```
Use "SametimeAgent"
```

Script libraries

Copy the following script libraries from the Sametime Toolkit Utilities database into the application database.

Script Name	*Script Function*
SametimeAgent	A script program used in the Sametime server login process. (The source code is removed from this script)
SametimeClient	A script program used in the Sametime server login process. (The source code is removed from this script)
SametimeHost	Scripts used in locating a person's Sametime server.
SametimeServerName	A script program used to locate the Sametime server associated with the application.
SametimeStrings	A set of translatable strings called by the Sametime script libraries.

Note We are not using the VP and VPStrings scripts in this instance as we did when addingWho is Here functionality using the LSX functions. The VP script is responsible for accessing, loading, and manipulating the ActiveX components on the user's workstation that were installed when the user installed the Connect Client. Since they are not used in this instance, it is not necessary to add them here.

Fields to add to a form

When creating a new form, there are several fields that need to be added in order for the Who is Here feature to work. The fields and their purposes are as follows:

Field Name	*Purpose*
User_Name	Identifies the name of the user.
HTTP_User_Agent	Identifies the browser the client is using.
IsMozila3	Identifies whether the correct release of Netscape Navigator or Internet Explorer is being used.
SAMETIME_TOKEN	Stores the user's token.
SametimeReplicaID	The Database Replica ID used to identify the database as a Place.
SametimeDocumentID	The Document ID used to identify a document as a Place.
SametimeServer	Identifies the Sametime server to log in.

These are the minimal fields that must be placed on the form in order for Sametime to function correctly. There may be other fields necessary to run any Who is Here applet. These additional fields should also be added. These fields should be of the datatype Computed for Display. This prevents the data in these fields from being stored.

A recommended way to add these fields to the form is to add them to a subform. That way you can manage them all in one place.

Agents in the WebQueryOpen event

In the WebQueryOpen event for the form, add the following code:

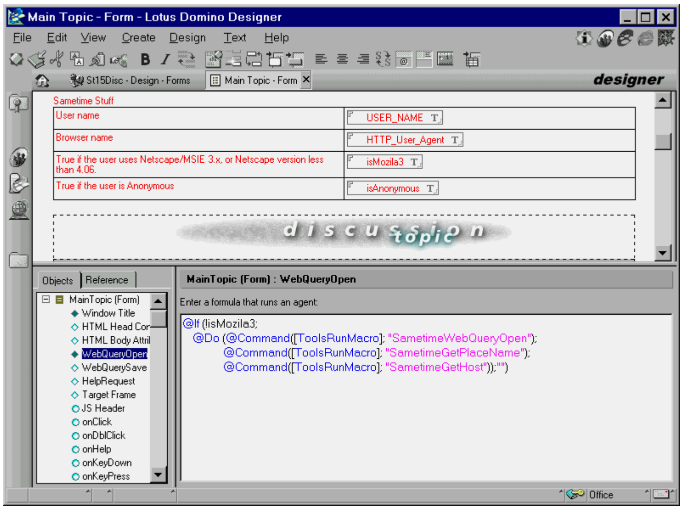
```
@If (!isMozila3;

  @Do (@Command([ToolsRunMacro]; "SametimeWebQueryOpen");

     @Command([ToolsRunMacro]; "SametimeGetPlaceName");

     @Command([ToolsRunMacro]; "SametimeGetHost”));"")
```

If you need to run other agents here, they can also be added to the list, and set to run after these agents run. Since the Who is Here functionality and other Sametime functions will run on Netscape Navigator 4.0.6 or higher or Microsoft Internet Explorer 4.x or higher, this will prevent errors when other browsers try to run the code. The field IsMozila3 is a Calculated for Display field, which has the following formula:

```
@Matches(@LowerCase(HTTP_User_Agent);
"*mozilla/3*|*mozilla/2*") |

@Matches(@LowerCase(HTTP_User_Agent);
"*mozilla/4.0{ 0-5}*&!*compatible*")
```

If the formula for the field IsMozila3 returns a False, then the agents needed to run in the WebQueryOpen event for the form will be set to not run and the user will not attempt to connect to the Sametime server.

In the following figure, we show the code for the Sametime 1.5 Discussion template.



Sametime Stuff

User name	USER_NAME T
Browser name	HTTP_User_Agent T
True if the user uses Netscape/MSIE 3.x, or Netscape version less than 4.06.	isMozila3 T
True if the user is Anonymous	isAnonymous T

MainTopic (Form) : WebQueryOpen

Enter a formula that runs an agent:

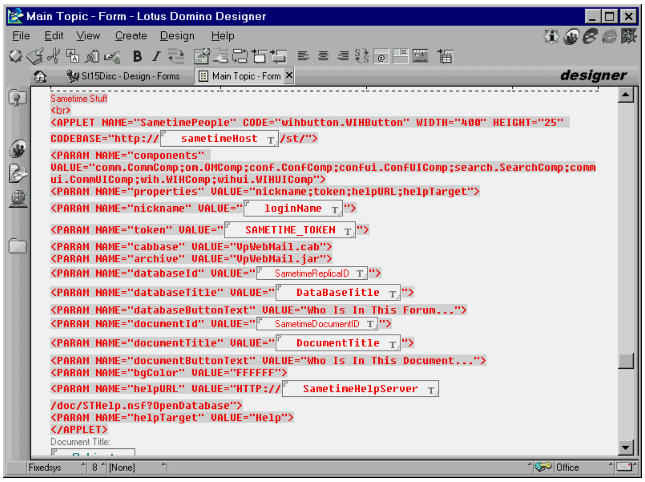
```
@If (!isMozila3;
  @Do (@Command([ToolsRunMacro]; "SametimeWebQueryOpen");
        @Command([ToolsRunMacro]; "SametimeGetPlaceName");
        @Command([ToolsRunMacro]; "SametimeGetHost"));"")
```

Once this is complete, you can add the Who is Here functionality to the database's documents.

Adding HTML to a form to load the applet

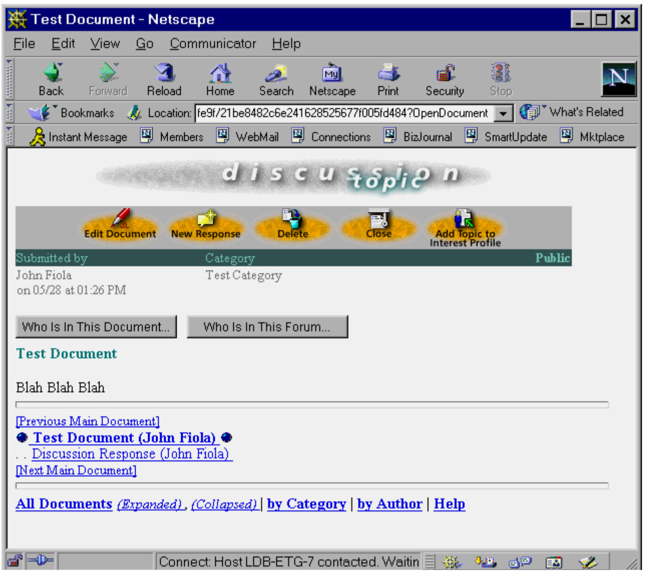
Add the HTML code which will load and run the Who is Here applet on the Notes form. Some of the parameters may be variable, such as the database ID, document ID, and the name of the Sametime server. You can create Compute for Display fields to load these parameters into the applet at runtime. By adding Compute for Display fields, the values are not stored with the document, allowing for minimizing the space used in the database. This also allows parameters to be different for both documents and users.

The following figure shows the HTML code used to load the Who is Here applet in the Discussion database that is supplied with the Sametime server.


Main Topic - Form - Lotus Domino Designer
File Edit View Create Design Help
St15Disc - Design - Forms
Main Topic - Form
designer

```
Sametime Stuff
<br>
<APPLET NAME="SametimePeople" CODE="wihbutton.WIHButton" WIDTH="400" HEIGHT="25"
CODEBASE="http:// sametimeHost /st/">
<PARAM NAME="components"
VALUE="comm.CommComp;om.OMComp;conf.ConfComp;confui.ConfUIComp;search.SearchComp;comm
ui.CommUIComp;wih.WIHComp;wihui.WIHUIComp">
<PARAM NAME="properties" VALUE="nickname;token;helpURL;helpTarget">
<PARAM NAME="nickname" VALUE=" loginName ">
<PARAM NAME="token" VALUE=" SAMETIME_TOKEN ">
<PARAM NAME="cabbase" VALUE="VpWebMail.cab">
<PARAM NAME="archive" VALUE="VpWebMail.jar">
<PARAM NAME="databaseId" VALUE=" SametimeReplicaID ">
<PARAM NAME="databaseTitle" VALUE=" DataBaseTitle ">
<PARAM NAME="databaseButtonText" VALUE="Who Is In This Forum...">
<PARAM NAME="documentId" VALUE=" SametimeDocumentID ">
<PARAM NAME="documentTitle" VALUE=" DocumentTitle ">
<PARAM NAME="documentButtonText" VALUE="Who Is In This Document...">
<PARAM NAME="bgColor" VALUE="FFFFFF">
<PARAM NAME="helpURL" VALUE="HTTP:// SametimeHelpServer
/doc/STHelp.nsf?OpenDatabase">
<PARAM NAME="helpTarget" VALUE="Help">
</APPLET>
Document Title:
```

When we open a document in the database, we see the following:



Submitted by | Category | Public
John Fiola on 05/28 at 01:26 PM | Test Category

Who Is In This Document... Who Is In This Forum...

Test Document

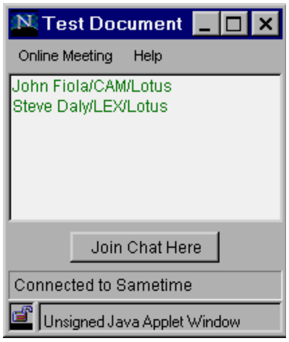
Blah Blah Blah

[Previous Main Document]
Test Document (John Fiola)
. . Discussion Response (John Fiola)
[Next Main Document]

All Documents (Expanded) , (Collapsed) | by Category | by Author | Help

Connect: Host LDB-ETG-7 contacted. Waitin

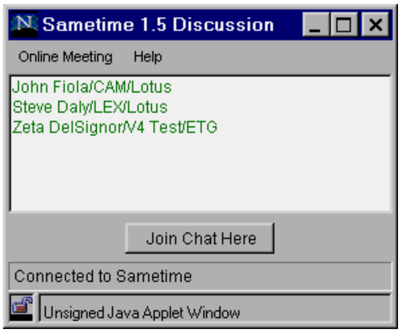
When we click the Who Is In This Document button, the following box opens:



John Fiola/CAM/Lotus
Steve Daly/LEX/Lotus

Connected to Sametime
Unsigned Java Applet Window

We see a similar dialog box when the Who Is In This forum button is clicked.



John Fiola/CAM/Lotus
Steve Daly/LEX/Lotus
Zeta DelSignor/V4 Test/ETG

Connected to Sametime
Unsigned Java Applet Window

Using JavaScript to load the applet

Instead of directly embedding HTML code on the form to load and run a Who is Here applet, we can use JavaScript to perform the same function.

Add functions to the JavaScript Header (JS Header) of the Main form:

```
var newFrame
```

This function sets up various parameters for later functions:

```
function authProf() {

var pathname = window.location.pathname;

 filename =
   pathname.substring(0,(pathname.lastIndexOf(“nsf”)+4))

 key = document.forms[0].AbrFrom.value
 var newWindow = window.open(filename +
         "LookupPersonalProfiles/" + key +
         "?OpenDocument&hw=1","secondary_window",
         "toolbar=no,location=no,scrollbars=yes,
          directories=no,height=500,width=625")
}
```

The function below converts spaces in a string to plus signs (+):

```
function spaceToPlus(param) {
 param = "" + param; // just to be sure this is a string
 var splittedParam = param.split(" ");
 param = "" + splittedParam[0];
 for (var i = 1; i < splittedParam.length; i++)
  param += "+" + splittedParam[i];
 return param;
}
```

The function below opens the Who is here window and runs the applet:

```
function openSametimeChatWindow()
{
 // Open the WIH Form in the Bottom Frame
 var databaseName = document.forms[0].DbName.value
 var placeids = document.forms[0].SametimeDocumentID.value
 var placenames = document.forms[0].DocumentTitle.value
 var nickname = document.forms[0].loginName.value
 var token = document.forms[0].SAMETIME_TOKEN.value
 var sametimehost = document.forms[0].sametimeHost.value
 var IsWebUser = document.forms[0].isWebUser.value

 if (IsWebUser == "N")
 {
   newFrame = window.open("/" + databaseName +
             "/SametimeLiveChat?OpenForm",
             "appletFrame");
 }
 else
 {
   var urlString = "/" + databaseName +
       "/SametimeWebLiveChat?OpenForm&placeids=" +
       placeids + "&placenames=" + placenames +
       "&nickname=" + nickname + "&token=" +
       token + "&sametimehost=" + sametimehost;

   urlString = spaceToPlus(urlString);

   newFrame = window.open(urlString, "appletFrame");

 }

}
```

Note The above code has been formatted for display on the page. Pay close attention to the line breaks.

This script does the following:

- Reads in the applet parameters from the document's form.
- Checks whether the user's client is a Notes client or a Web browser.
- Uses the appropriate way to load the applet. In the case of the Web browser, it does the following:
  - Creates a URL string to open a form called SametimeWebLiveChat (this is created next) and to pass the form a set of parameters.
  - Converts any spaces in the URL string to plus signs (+).
  - Opens the form SametimeWebLiveChat (this is the name of the separate form we will create next) that contains the applet in the new frame.

Next, create a new form in your database. In this case, we call it SametimeWebLiveChat.

In the JavaScript Header (JS Header) section of this form we add some JavaScript to parse the parameters that are submitted from the URL call above, which then loads the who is here applet by printing out the HTML needed to load the applet.

This function converts a plus sign (+) to a space for a given string parameter:

```
function plusToSpace(param) {

 param = "" + param; // just to be sure this is a string

 var splittedParam = param.split("+");

 param = "" + splittedParam[0];
 for (var i = 1; i < splittedParam.length; i++)
  param += " " + splittedParam[i];
 return param;
}
// Create Applet Code using parameters


var params = location.search.substring(1,
                 location.search.length);
 if (params.length != 0) {
  params = unescape(params);
  params = plusToSpace(params);
}
```

The following lines extract the specific parameters from the URL which loads the form:

```
// extract params
var placeids = extractArgument(params, "placeids");
var placenames = extractArgument(params, "placenames");
var nickname = extractArgument(params, "nickname");
var token = extractArgument(params, "token");
var scodebase = extractArgument(params, "sametimehost");
```

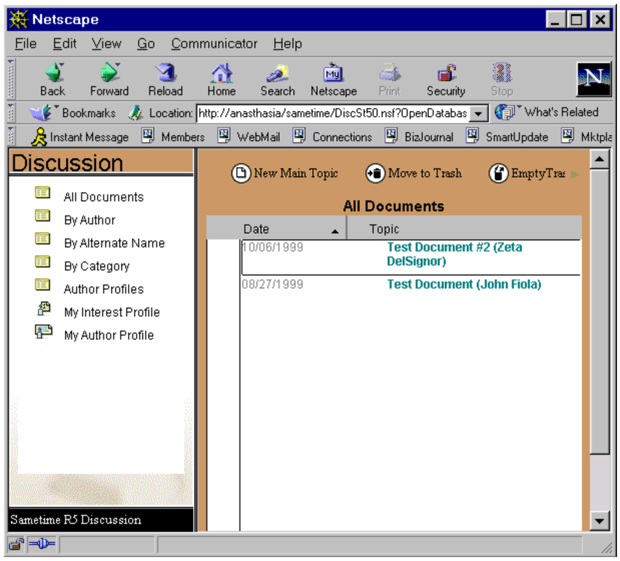
The following lines print out to the document the HTTP code used to run the applet:

```
document.writeln('<HTML>');
document.writeln('<BODY BGCOLOR="LightGrey">');
document.writeln('<APPLET CODEBASE="http://', scodebase,
'/sametimeapplets/" CODE="DiscusChatApplet" WIDTH="100%"
HEIGHT="100%">');
document.writeln('<PARAM NAME="archive"
VALUE="VpApi.jar,DiscusChat.jar">');
document.writeln('<PARAM NAME="cabbase"
VALUE="VpApi.cab,DiscusChat.cab">');
document.writeln('<PARAM NAME="nickname" VALUE="',
nickname, '">");
document.writeln(’<PARAM NAME="token" VALUE="', token,
'">');
```

```
document.writeln('<PARAM NAME="placeids" VALUE="',
placeids, '">');

document.writeln('<PARAM NAME="placenames" VALUE="',
placenames, '">');

document.writeln('<PARAM NAME="bgcolor"
VALUE="0xc0c0c0">');

document.writeln('<PARAM NAME="rightoffset" VALUE="0">');

document.writeln('<PARAM NAME="bottomoffset" VALUE="0">');

document.writeln('<!-- I18N -->');

document.writeln('<PARAM NAME="send_button_label" VALUE="
Send ">');

document.writeln('<PARAM NAME="conf_invitation"
VALUE="Please, join my chat now!">');

document.writeln('<PARAM NAME="conf_name" VALUE="My cool
conference.">');

document.writeln('<PARAM NAME="chat_menu_label"
VALUE="Message...">');

document.writeln('<PARAM NAME="sametimeserver" VALUE="',
scodebase, '">');

document.writeln('<PARAM NAME="statuslabel" VALUE=" ">');

document.writeln('<PARAM NAME="namelabel" VALUE="Name">');

document.writeln('<PARAM NAME="show_status_icon"
VALUE="true">');

document.writeln('<PARAM NAME="status_active" VALUE="I Am
Active">');

document.writeln('<PARAM NAME="status_not_using"
VALUE="Not using computer">');

document.writeln('<PARAM NAME="status_away" VALUE="I Am
Away">');

document.writeln('<PARAM NAME="status_dnd" VALUE="Do Not
Disturb Me">');

document.writeln('<PARAM NAME="status_unresolved"
VALUE="Unresolved">');

document.writeln('<PARAM NAME="status_offline"
VALUE="Offline">');

document.writeln('<PARAM NAME="show_status_description"
VALUE="false">');
```

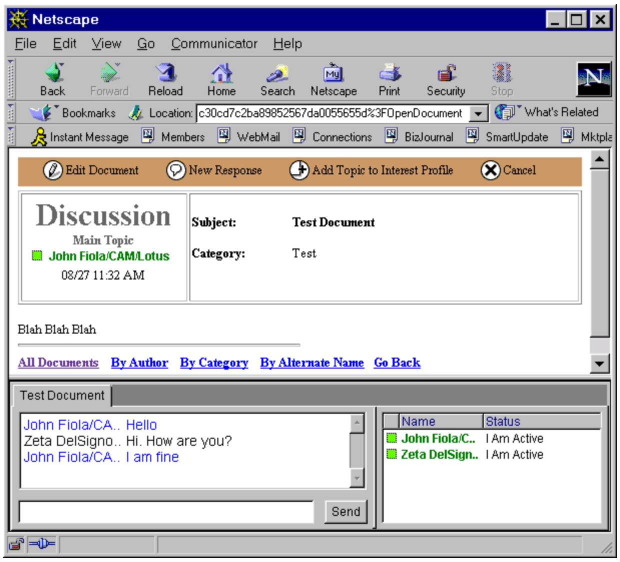
```
document.writeln('</APPLET>');
document.writeln('</BODY>');
document.writeln('</HTML>');


/**
 * Given a string of the form
name1=value1&name2=value2&...,
 * extracts value of the coupld name=value by name.
 */
function extractArgument(params, name) {
 var ix = -1;
 var iy = -1;
 if (params.length != 0) {
  args = params.toLowerCase();
  arg = name.toLowerCase() + "=";
  ix = args.indexOf(arg);
  if (ix != -1) {
   ix += arg.length;
   iy = args.substring(ix, args.length).indexOf("&");
   if (iy == -1)
    iy = args.length;
   else
    iy += ix;
  }
 }
 return ix != -1 ? (ix < iy ? params.substring(ix, iy) :
"") : null;
}
```

When we open a document in the database, we see the following:


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://anasthasia/sametime/DiscSt50.nsf?OpenDatabas What's Related
Instant Message Members WebMail Connections BizJournal SmartUpdate Mktpla
Discussion
All Documents
By Author
By Alternate Name
By Category
Author Profiles
My Interest Profile
My Author Profile
Sametime R5 Discussion
New Main Topic Move to Trash EmptyTras
All Documents

Date	Topic
10/06/1999	Test Document #2 (Zeta DelSignor)
08/27/1999	Test Document (John Fiola)

When we open a document, a frame opens at the bottom and an applet is loaded in the bottom frame showing who is in the document.


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: c30cd7c2ba89852567da0055655d%3FOpenDocument What's Related
Instant Message Members WebMail Connections BizJournal SmartUpdate Mktpla
Edit Document New Response Add Topic to Interest Profile Cancel
Discussion
Main Topic
John Fiola/CAM/Lotus
08/27 11:32 AM

Subject: Test Document

Category: Test

Blah Blah Blah

All Documents By Author By Category By Alternate Name Go Back

Test Document

```
John Fiola/CA.. Hello
Zeta DelSigno.. Hi. How are you?
John Fiola/CA.. I am fine
```

Name	Status
🟩 John Fiola/C..	I Am Active
🟩 Zeta DelSign..	I Am Active

The frame at the lower right corner shows who is reading the document and their online status. The frame on the lower left shows the chat conversation that is going on among the readers of the document. The top section of this frame shows the conversation thread. The bottom is for the user to enter text if they wish to participate in the conversation.

Chapter 6
Using the Meeting Services API

The Meeting Services API provides methods of controlling Sametime meetings and customizing the user interface of the Sametime Meeting applet. With this API we can customize the applet that is contained in the Sametime Online Meeting Center. Or, we can wrap the meeting applet into our own Java applet for use in other Domino applications.

In this chapter we will construct a Domino R5 application that will use two applets built with the Meeting Services API, after first reviewing the setup of our development environment.

Important The Domino application and the Java source files used in this chapter can be found on the IBM Redbooks Web site, **`http://www.redbooks.ibm.com`**

Setting up the Domino environment

The application described in this chapter will be hosted by a Domino R5 server. We will also need a Sametime 1.5 server on the network to host the meetings that we are going to create.

As we discussed in Chapter 2, Sametime-enabled applications do not have to be hosted by the Sametime server, but you will need to make sure that a replica of the Secrets and Tokens databases is located on the same server on which your Sametime-enabled application resides. The Secrets and Tokens databases are used by Sametime for authentication, which is described in detail in Chapter 2, Java development.

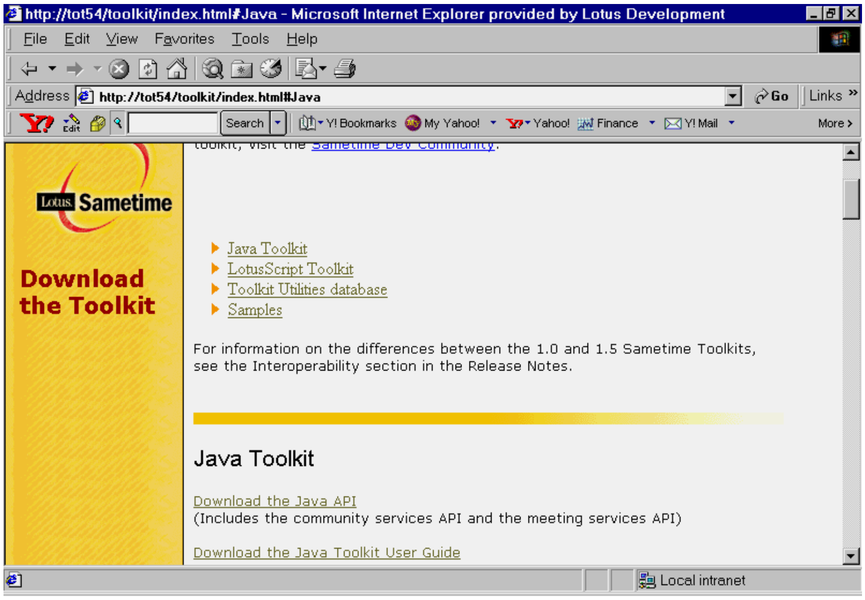
Before building the applets, we first need to download and install a Java development toolkit into our development environment.

You can use any Java development tool that you prefer. We have chosen to use the Sun JDK 1.1.8.

Important The minimum release level of JDK for building applets with the Meeting Services API is JDK 1.1.

Getting the toolkit

The Meeting Services API Toolkit is copied to the Sametime server during installation. It can be found on the Sametime Toolkit page on the Sametime server.


http://tot54/toolkit/index.html#Java - Microsoft Internet Explorer provided by Lotus Development
File Edit View Favorites Tools Help
Address http://tot54/toolkit/index.html#Java
Lotus Sametime
Download the Toolkit
Local intranet

toolkit, visit the Sametime Dev Community.

- Java Toolkit
- LotusScript Toolkit
- Toolkit Utilities database
- Samples

For information on the differences between the 1.0 and 1.5 Sametime Toolkits, see the Interoperability section in the Release Notes.

Java Toolkit

Download the Java API
(Includes the community services API and the meeting services API)

Download the Java Toolkit User Guide

Installing the toolkit

Once you download the toolkit, unzip it to your development directory. The zip file contains the following files that will be used for developing Meeting Services applications:

File	*Purpose*
STMeetingApplet.jar	Sametime Meeting Applet files archived in .JAR format
STMeetingApplet.cab	Sametime Meeting Applet files archived in .CAB format
IMAGES folder	.GIF files used by the Meeting Applet
PROPERTIES folder	Property files for internationalization of the Meeting Applet

The toolkit also includes the Community Services Java API, as well as some sample Java source code files. The Sametime server installation also contains several example Sametime applications.

STMeetingApplet.jar contains the Sametime Meeting Applet and all the supporting class files. It must be available to the Java compiler and be included in the package with your custom applet. The contents of the IMAGES and PROPERTIES folder should also be included with your custom applet.

Next, add the STMeetingApplet.jar to your CLASSPATH environment variable so that the Java compiler can find the Sametime Meeting Services class files. We unzipped the toolkit to the c:\jdk1.1.8\work directory. To set CLASSPATH enter the following:

```
Set CLASSPATH=<other>;c:\jdk1.1.8\work\STMeetingApplet.JAR
```

Your class code must import the contents of STMeetingApplet.jar:

```
import com.databeam.sametime.STMeetingApplet
```

Building a Domino application

For this example, we built a very simple Domino application called the Huddle Room. The Huddle Room is used by a group of people who need to meet on a topic and discuss some ideas. They don't need something as formal as the Sametime Online Meeting Center.

Copying the Sametime components

There are several components necessary to Sametime-enable a Domino application. These components are required to successfully authenticate with the Sametime server. You can find these components in the Sametime Online Meeting Center.

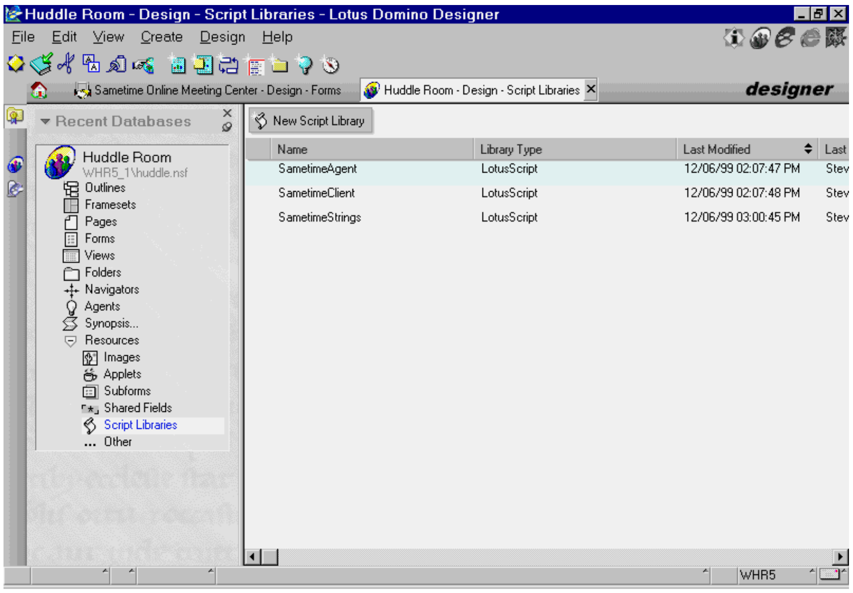
Note If you are adding Meeting Services to an application that already has Community Services, there is no need to recopy these components to your application.

Important The source code for these Sametime components has been hidden for security reasons.

Sametime script libraries

Copy the corresponding script libraries to your Domino application:

Script Library Name	*Purpose*
SametimeAgent	Used by the SametimeWebQueryOpen agent for Web users
SametimeClient	Used by the form that contains the Sametime applets
SametimeStrings	Used by the other Sametime components for strings used in messages and dialog boxes

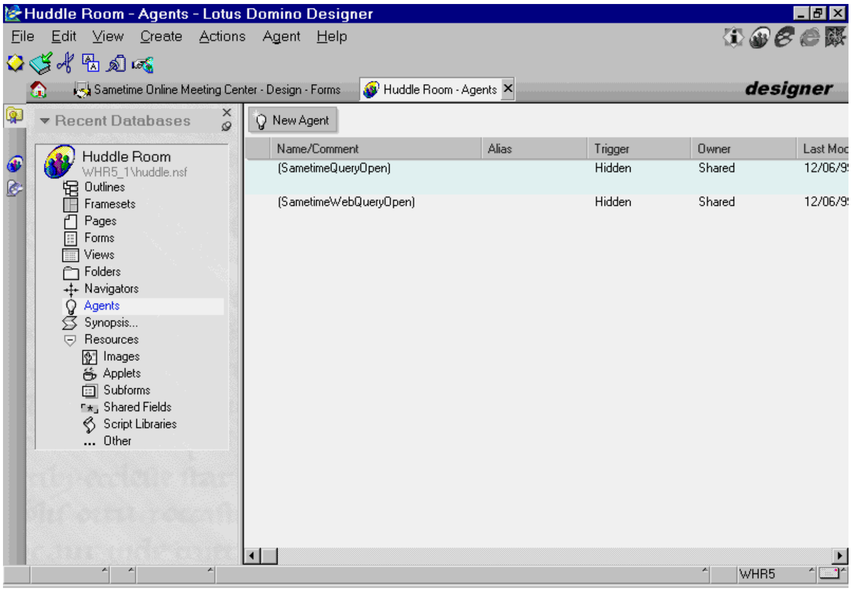


Name	Library Type	Last Modified	Last
SametimeAgent	LotusScript	12/06/99 02:07:47 PM	Stev
SametimeClient	LotusScript	12/06/99 02:07:48 PM	Stev
SametimeStrings	LotusScript	12/06/99 03:00:45 PM	Stev

Sametime agents

As we stated in the previous section, in order to connect to the Sametime server we need to get a token. This token is created by agents that run on the Sametime server:

Agent Name	*Purpose*
SametimeQueryOpen	Used by Notes clients to retrieve a token
SametimeWebQueryOpen	Used by Web clients to retrieve a token

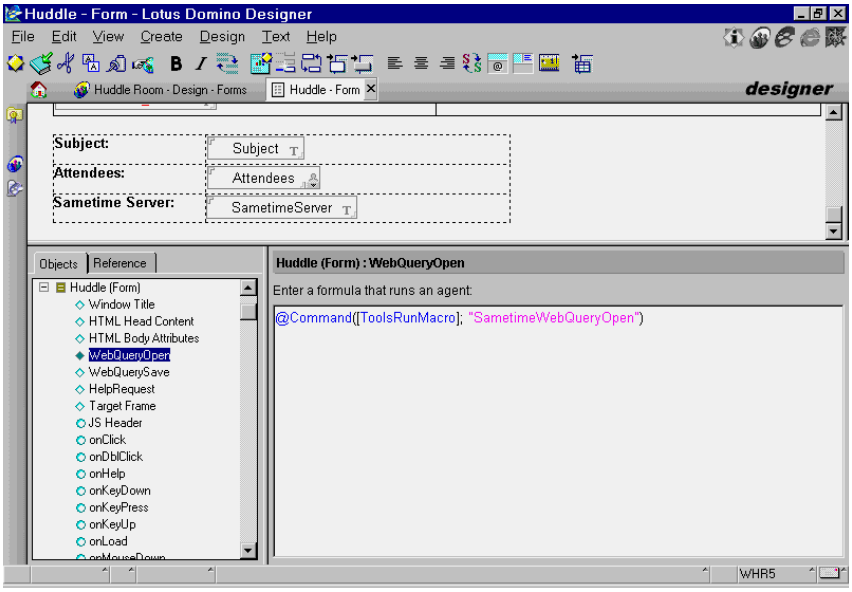


Name/Comment	Alias	Trigger	Owner	Last Moc
(SametimeQueryOpen)		Hidden	Shared	12/06/9
(SametimeWebQueryOpen)		Hidden	Shared	12/06/9

Designing the form

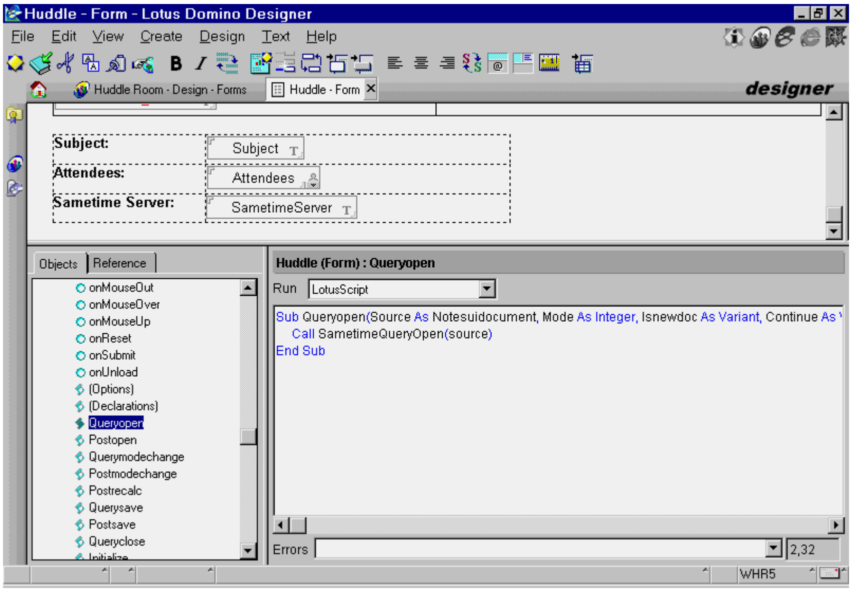
Getting the authentication token

We must call the appropriate Sametime agent to get an authentication token for the Sametime server. The SametimeWebQueryOpen is called in the WebQueryOpen event of the form.



```
@Command([ToolsRunMacro]; "SametimeWebQueryOpen")
```

To support Notes clients as well as browsers we add a USE statement in the Globals section of the form. We then add a call to the SametimeQueryOpen function in the QueryOpen event passing the NotesUIDocument object of the current document as an argument.



```
Sub Queryopen(Source As Notesuidocument, Mode As Integer, Isnewdoc As Variant, Continue As
    Call SametimeQueryOpen(source)
End Sub
```

Adding the fields

Java applets built with the Meeting Services API will work in any Domino form or subform. However, there are a few requirements for operation.

First, to get a proper authentication token, your form must have two fields. They can either be hidden or visible, but they must be named the correct way for the token to be generated. Here are the fields and their composition:

Field Name	*Composition*
USER_NAME	Computed for Display text field. Formula is @Username
SAMETIME_TOKEN	Editable text field. Default formula is SAMETIME_TOKEN

Second, you must run the SametimeWebQueryOpen agent (for Web clients) or SametimeQueryOpen (for Notes clients) in the form's WebQueryOpen or QueryOpen events. These agents use the above fields to create and pass the proper token to the client. The contents of these fields are then passed to the applets.

The rest of the form is made up of fields for the huddle title and a field for the name of the Sametime server where the meeting will be held.

The rest of the application

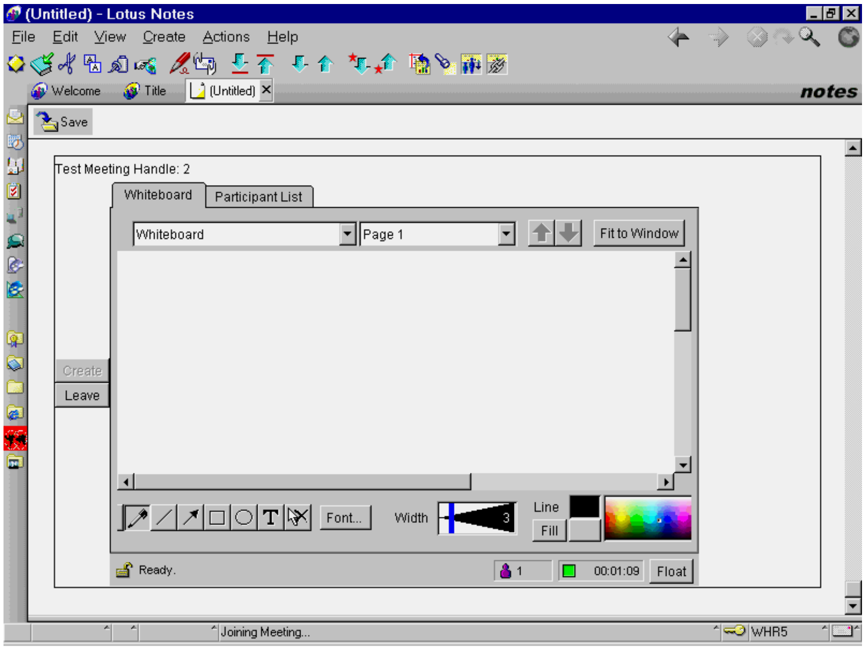
We created a navigation outline, a couple of pages, and a frameset for the infrastructure of our application.

Building the Huddle Manager applet

The Huddle Manager applet will be used by the person who called the huddle. It will have the following functionality:

- Create and join a Sametime meeting
- Leave a Sametime meeting
- Provide a whiteboard for creating drawings
- List the participants in the huddle

The following figure shows the Manager applet in action.



The Huddle Manager applet takes four parameters from the Domino application:

Parameter Name	*Purpuose*
MeetingTitle	Name of the meeting. Displayed at the top of the applet.
User_Name	The user's name. Needed for authentication. Displayed in the Participant List viewer.
Sametime_Token	Authentication token provided by the appropriate agent.
Sametime_Server	Name of the Sametime server that the user must log in to for this meeting.

These parameters will be passed to the applet via the HTML APPLET tag that is created by Domino.

Getting started

The first order of business is to declare and instantiate all the variables and classes that we are going to use. The following code shows the class declaration and the init() method of our applet.

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

import com.databeam.sametime.STMeetingApplet;

public class HuddleMgrApplet extends Applet implements
AppletStub, ActionListener {

     STMeetingApplet meetingApplet;
     boolean meetingAppletStarted;
     Label meetingTitleLabel;
     Button createMeeting;
     Button leaveMeeting;
     int meetingHandle;
     GridBagConstraints constraints = new
          GridBagConstraints();

  String meetingTitle;
  String meetingUserName;
String meetingToken;
String meetingServer;

    public void init() {

          meetingTitle = getParameter("MeetingTitle");
          meetingUserName = getParameter("User_Name");
          meetingToken = getParameter("Sametime_Token");
          meetingServer = getParameter("Sametime_Server");

          meetingApplet = new STMeetingApplet();
          meetingApplet.setStub(this);
          meetingApplet.addParameter("IsMeetingManager”,"1");

          meetingAppletStarted = false;
          meetingTitleLabel = new
                Label(meetingTitle.toString());
          createMeeting = new Button("Create");
          leaveMeeting = new Button("Leave");
          meetingHandle = 0;

          Panel p = new Panel(new GridBagLayout());
          constraints.fill = GridBagConstraints.HORIZONTAL;
          constraints.gridx = 1;
          constraints.gridy = 1;
          p.add(createMeeting, constraints);
          constraints.gridy = 2;
          p.add(leaveMeeting, constraints);

          setLayout( new BorderLayout() );
          add("North", meetingTitleLabel);
          add("West", p);
          add("Center", meetingApplet);
```

```
            createMeeting.setEnabled(true);
            leaveMeeting.setEnabled(false);

            createMeeting.addActionListener(this);
            leaveMeeting.addActionListener(this);
        }
```

Wrapping the meeting applet

The Meeting Services API allows you to customize the existing meeting applet that is used in the Sametime Online Meeting Center. Your custom applet is actually an applet viewer, or "wrapper," for the meeting applet. We tie the two applets together via the Java AppletStub class. This is done in the init() method for the class.

The next procedure is to set up the meeting applet via parameters. Parameters are set via the addParameter() method. In the code above, we are setting the IsMeetingManager to "1". This tells the Sametime server that the user who uses this applet is the moderator of the meeting. Some other parameters can also be set:

Parameter Name	*Definition*
ConferenceHandle	Handle to a Sametime meeting.
IsFloatable	Indicates if the meeting applet can be floated outside the browser as a separate window. "0" to not allow. "1" to allow.
AutoFloat	Indicates whether the meeting applet should be floated when it is started. "0" to not float. "1" to float.

Note The parameters for the meeting applet will change from version to version of Sametime, further exposing functionality. Please refer to the API documentation that came with the toolkit for a complete list of parameters.

Creating the meeting

We want to create a meeting on the Sametime server when the manager clicks the Create button. We've implemented this as an ActionListener in the applet. When you click the button, the following code runs to create the meeting:

```
meetingHandle = meetingApplet.createMeeting(0, meetingTitle,
meetingUserName, meetingToken, tokenType, meetingServer, 8081,
8081, false);
```

The createMeeting() function creates a meeting on the Sametime Server and returns the meeting's handle. In our applet, many of the parameters for the createMeeting() function are passed as parameters to the applet. You could also modify the code to pull the information from other Domino or back-end sources.

The following items are elements of the createMeeting() function:

Parameter Name	*Definition*
handle	Integer. Handle to the Sametime meeting. If "0," a new meeting will be created.
name	String. Name of the Sametime meeting.
userName	String. User's name.
accessToken	String. Authentication token.
accessTokenType	Byte. Type of authentication token. Should always be set to "1."
host	String. Sametime host name.
port	Integer. TCP/IP port used to connect to the Sametime server.
tunnelPort	Integer. TCP/IP port used to make an HTTP tunneling connection to the Sametime server.
meetingEncrypted	Boolean. True if you want the meeting data encrypted. False if not.

Joining the meeting

If the meeting was created successfully, the Sametime server returns a meeting handle. The next step is to join the meeting using this handle as a parameter. The joinMeeting() method joins an existing meeting on the Sametime server. The method has two parameters:

Parameter Name	*Definition*
meetingHandle	Integer. The handle to the Sametime meeting. Returned from the createMeeting() method.
maxWaitTime	Integer. Maximum amount of time to wait for the meeting.

Here is the code to join the meeting.

```
if (meetingApplet.joinMeeting(meetingHandle, 0) == 1) {
      meetingApplet.addViewer(
            "com.databeam.draw.v15DBDrawPanel",true);
      meetingApplet.addViewer(
"Com.databeam.meetingmanager.v15MeetingManagerPanel"
          ,false);

     meetingTitleLabel.setText(meetingTitleLabel.getText() +
"         Handle: " + String.valueOf(meetingHandle));
}
```

Note The last line of the code displays the handle of the meeting. This is important. If another user wants to join the meeting, they need to know the meeting handle. We display it in the user interface so the meeting manager can communicate it to the other users who need to attend the meeting.

Displaying the viewers

The meeting applet has several viewers that can be displayed. A viewer is added to the applet via the addViewer() method or the addParameter() method. The table below describes each of the viewers.

Viewer Name	*Purpose*
com.databeam.sametime.v15AppSharePanel	Displays content created by the AppShare control.
com.databeam.sametime.v15DBDrawPanel	Used to create or view content in the whiteboard area of the meeting.
com.databeam.sametime.v15MeetingManagerPanel	Contains controls needed to view participants and manipulate their roles in the meeting.

In our example code, we used the addViewer() method to add viewers to the meeting applet. The method has two parameters:

Parameter Name	*Definition*
viewerName	String. Name of the viewer to be added to the meeting applet.
isFocus	Boolean. Should the viewer be given focus once it has been added to the meeting applet.

Leaving the meeting

Now we have to add a function for the meeting manager to leave the meeting. We'll add another button to the user interface and modify the event handler. The following is the code to leave the meeting.

```
if (event.getSource() == leaveMeeting) {
     if (meetingAppletStarted == true) {
          this.showStatus("Leaving Meeting...");
          meetingApplet.leaveMeeting();
          meetingApplet.stop();
          meetingAppletStarted = false;
          createMeeting.setEnabled(true);
          leaveMeeting.setEnabled(false);
          meetingHandle = 0;
     }
}
```

The leaveMeeting() method removes the user from the Sametime meeting. In the case of the meeting manager, we also want to stop the meeting applet and reset the buttons to the correct state. Finally, we reset the meeting handle variable back to zero.

Cleaning up

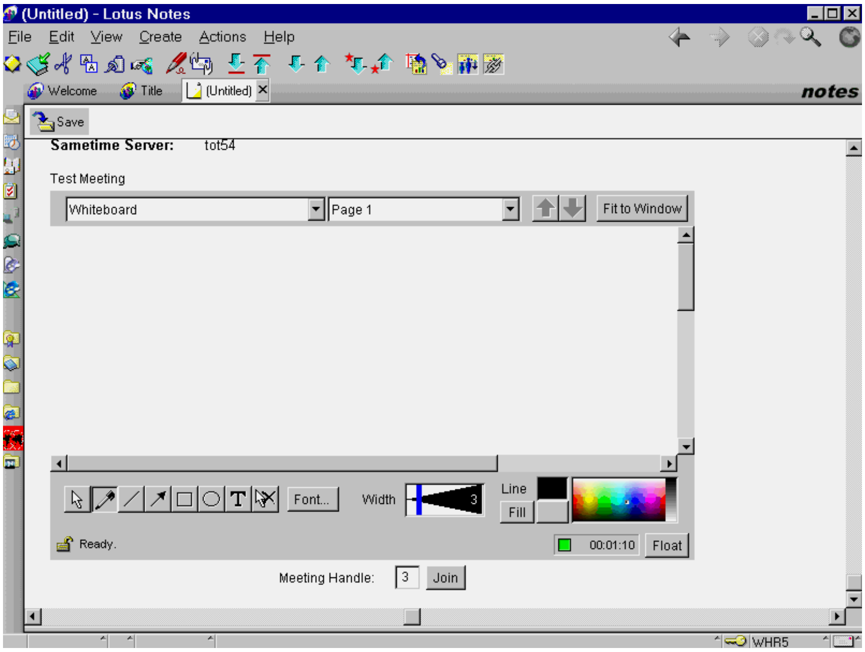
When the meeting manager closes the browser, or moves to another page, we want to leave the meeting and shut down the meeting applet. We also need to clean up the applet by running the destroy() method.

```
public void destroy() {
     if (meetingApplet != null) {
          if (meetingAppletStarted) {
               meetingApplet.leaveMeeting();
          }
          meetingApplet.stop();
          meetingApplet.destroy();
          meetingApplet = null;
     }
}
```

Building the client applet

The client applet will be used by people who attend huddle meetings. The applet is comprised of a whiteboard pane and a field to enter the meeting handle.

The client applet is very similar to the manager applet in form, but we do not include the participant viewer.



Sametime Server: tot54

Test Meeting

Ready.

00:01:10

Meeting Handle: 3

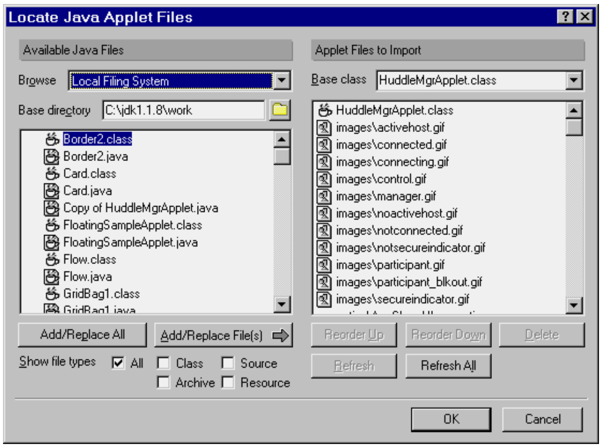
To join the meeting, the client needs to know the handle of the meeting. The meeting manager must provide this in some form. The client then enters the meeting handle number in the field and clicks the Join button.

```
String strMeetingHandle = inputMeetingHandle.getText();
meetingHandle = Integer.valueOf(strMeetingHandle).intValue();
meetingHandle = meetingApplet.createMeeting(meetingHandle,
     MeetingTitle, meetingUserName, meetingToken, tokenType,
     meetingServer, 8081, 8081, false);
if (meetingHandle != 0) {
     if(meetingApplet.joinMeeting(meetingHandle, 0) == 1) {
          MeetingApplet.addViewer(
           "com.databeam.draw.v15DBDrawPanel", true);
          meetingAppletStarted = true;
     }
}
```

We then take the meeting handle that was entered and call createMeeting(). This sets up the meeting applet for attending a Sametime meeting. If successful, the user then joins the meeting and adds the whiteboard viewer.

Importing the applets into the Domino application

To complete our application, we need to import the compiled Java classes, the STMeetingApplet.jar file and the images and property files. You could package all the files together and import the archive as a whole.



For the huddle application, we created two subforms: one for the meeting manager and the other for the attendees. We then set up a computed subform and tested whether the user was the meeting manager or an attendee.

The node controller

You can add more than one instance of the meeting applet. For example, you might have one panel of your applet set to show only the participant list viewer. Another panel could show the whiteboard viewer. One of the applets must be set as the node controller. The node controller is the applet responsible for maintaining certain operations of a Sametime meeting.

An example applet that uses the node controller functionality is included with the toolkit.

Where to go from here

You can use the applets that we created here as a good starting point for your applications. Among the enhancements you can make to this application are the following:

- Use two instances of STMeetingApplet within your applet, showing the participant list viewer in one instance and the whiteboard viewer in the other. Make sure to set one instance as the node controller.
- Add some CORBA functionality to write the meeting information back to the Domino database.

Compile the applets as Java applications, thus removing the need to be hosted by a Web server.

Chapter 7
Case study

The Millenia Company is an electronics retailer that has been in existence since 1985. Started in Cleveland, Ohio, by two brothers, Mark and Jeremy Jamison, the retailer specializes in entertainment electronics. Millenia grew rapidly, and four other stores were opened in Seattle, Boston, Atlanta, and San Francisco by 1990.

Since its opening, Millenia has attracted loyal customers who like the festive atmosphere of the stores and the personal attention of the salespeople. Millenia's prices are nationally competitive and their after-sales service has been rated among the best.

In the early years, Millenia's sales outpaced their closest competitors by 30 percent. They were riding the wave of America's interest in big screen TVs, DVD players, high speed VCRs and other popular electronics.

The state of Millenia

Millenia's booming sales were not an accident. The company's Marketing department enticed the electronics market from all angles. With a central Marketing department in Cleveland leading the way, Millenia advertised heavily in it local markets via television, print, and radio. The company name was well known, and favorable comments from satisfied customers did much to boast sales.

Until late 1996, Millenia's marketing strategy worked fine. At that time, however, management began to see a steady decline in sales. Millenia's customers were fleeing to the competition, pricing and profits were dropping, and sales objectives were not being met. More competitors coupled with an explosion of new technologies was causing Millenia huge losses in the marketplace. These factors drove Millenia's management to focus on new strategies for retaining its customers and for growing sales.

New approach

Millenia looked to the Internet to help solve its dilemma. With the advent of the Web and e-business, retailers had been thrown into the greatest marketing and selling event of the century. There was an explosion of new technology-driven channels to reach the customer. In this environment, competitors had to know what their customers wanted and give it to them online along with good service. Millenia needed to establish a strong presence in this arena if it was to keep pace with other savvy retailers.

By January 1997, Millenia had joined the online market. The company offered 75 percent of its products online at its user-friendly Web site. Sales from the site have been impressive since it was first launched, largely because of a greater market reach. By year end, however, customers were demanding more from the Web site. They wanted online assistance and a way to interact with customer service representatives. The lack of this convenience was driving customers to competitor's sites.

Millenia realized that service differentiation is one area where it is possible to lock in a customer and a competitive advantage. Management knew that the experience customers have as they interact with a company becomes a huge portion of the company's product. Strengthening their online customer relationship strategy could result in a number of quantitative benefits, including greater ability to up-sell and cross-sell, improved customer retention, and reduced cost of service. To accomplish this, Millenia needed to implement new processes to reach their customers on a new level.

Millenia's solution

To add a personal customer service element to its online market, Millenia introduced the WebProductCenter. This Java-based application uses the Web to connect its customer service representatives with online customers. The WebProductCenter is linked to the company's product databases, putting product information at the fingertips of customer service personnel.

At Millenia's WebProductCenter, customer service representatives interact with customers using text chat and online meetings. Customer service representatives can also forward Web pages containing specific product information to customers. These forms of instant communication make it easier and faster for customer service representatives to answer customer's questions. The process is not only efficient and less costly, but information provided to customers is more accurate.

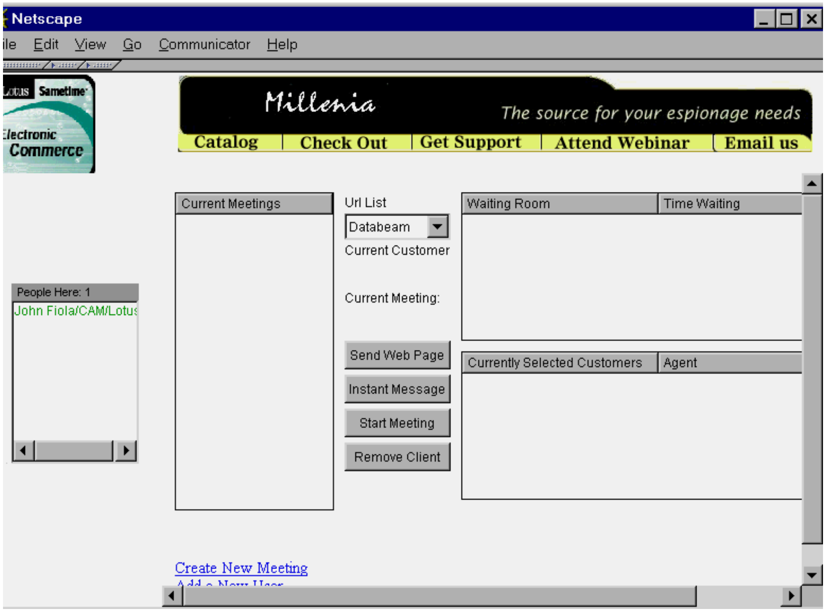
Almost everything that was accomplished with the WebProductCenter site was invisible to customers. With Lotus Sametime at the center of the application, a turnaround team added the WebProductCenter to the company's existing Web site within three weeks. As a result of the increased customer support, the company recovered from $1 million per year losses to $1.5 million per year profits within 12 months.

The rest of this case study explains the two components of Millenia's WebProductCenter, the service representative and the customer.

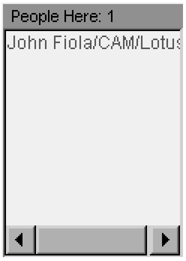
The service representative side

User interface

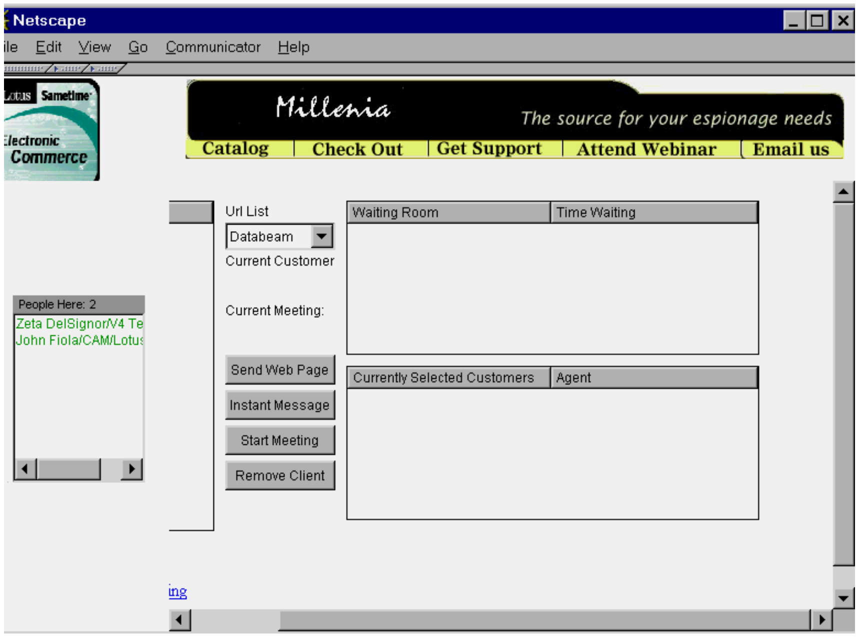
When the service representative opens the database, they have a control panel that looks like the following figure:



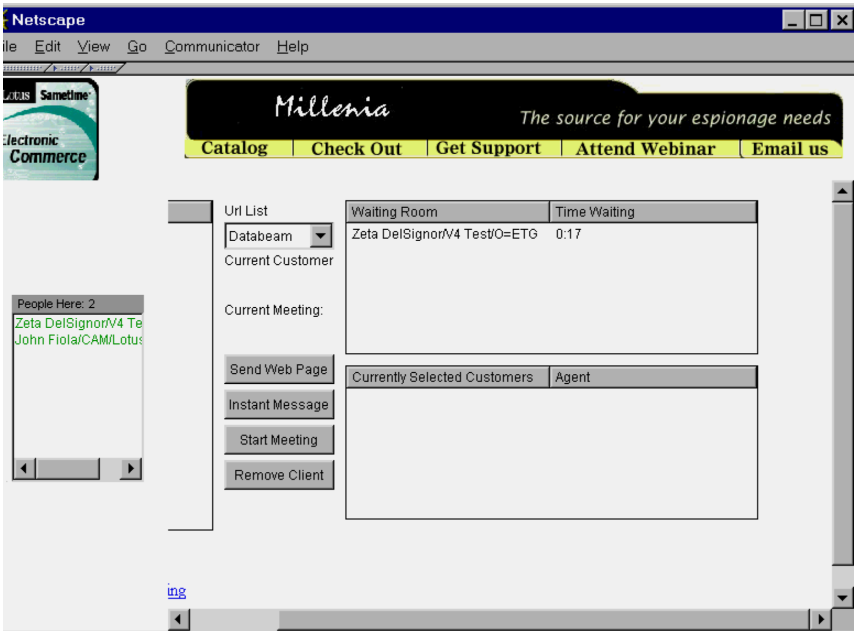
In the bottom left corner, there is a list of people who are logged in to the site. This shows all service representatives and customers who have logged in to the site.



Here we see that a customer has logged into the site:



The box in the top right corner, identified as the Waiting Room, shows a list of customers who have requested help from a service representative. It lists the name of the person and the amount of time the customer has been waiting for help.

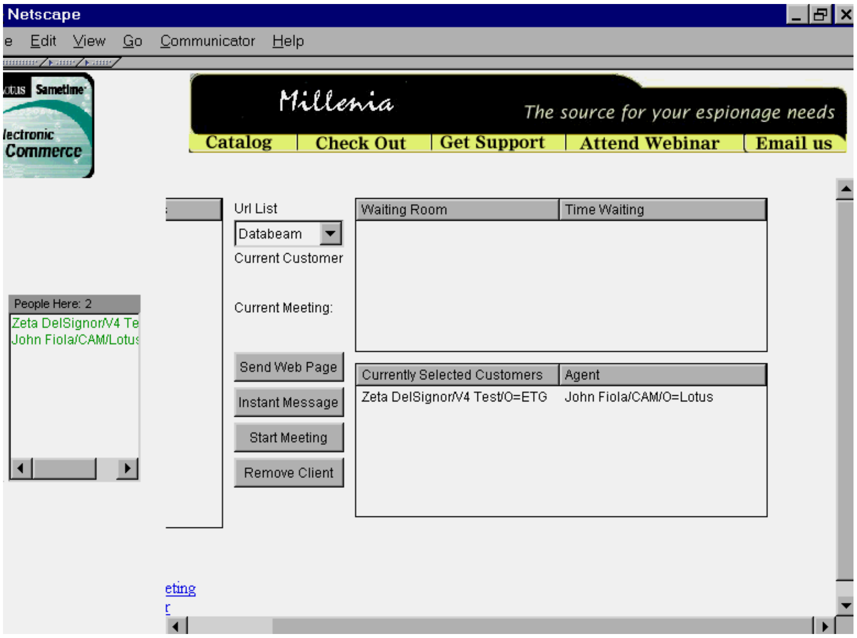


People Here: 2
Zeta DelSignor/V4 Te
John Fiola/CAM/Lotus

Waiting Room	Time Waiting
Zeta DelSignor/V4 Test/O=ETG	0:17

Currently Selected Customers	Agent

The service representative selects a person in the Waiting Room and that person is moved into the box on the lower right, identified as the Currently Selected Customer area.



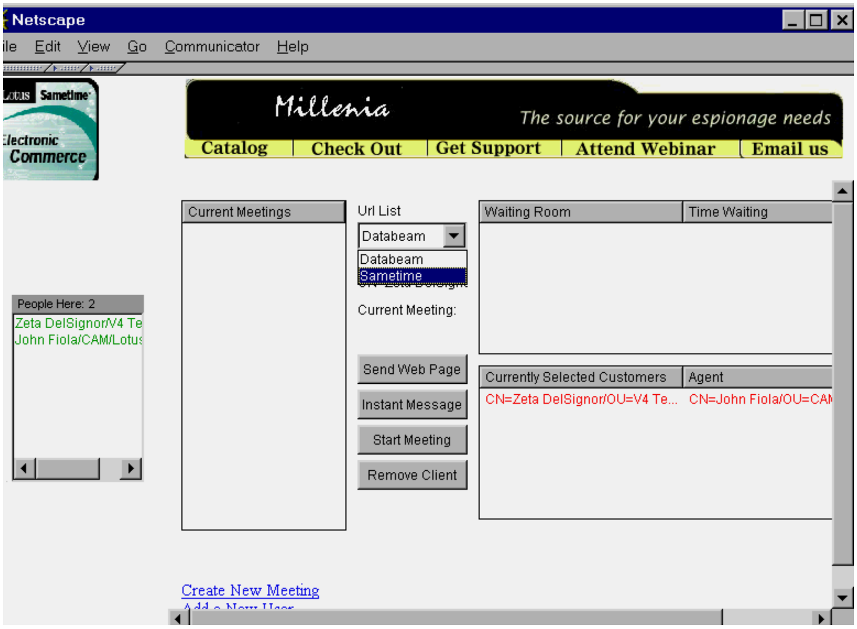
People Here: 2
Zeta DelSignor/V4 Te
John Fiola/CAM/Lotus

Waiting Room	Time Waiting

Currently Selected Customers	Agent
Zeta DelSignor/V4 Test/O=ETG	John Fiola/CAM/O=Lotus

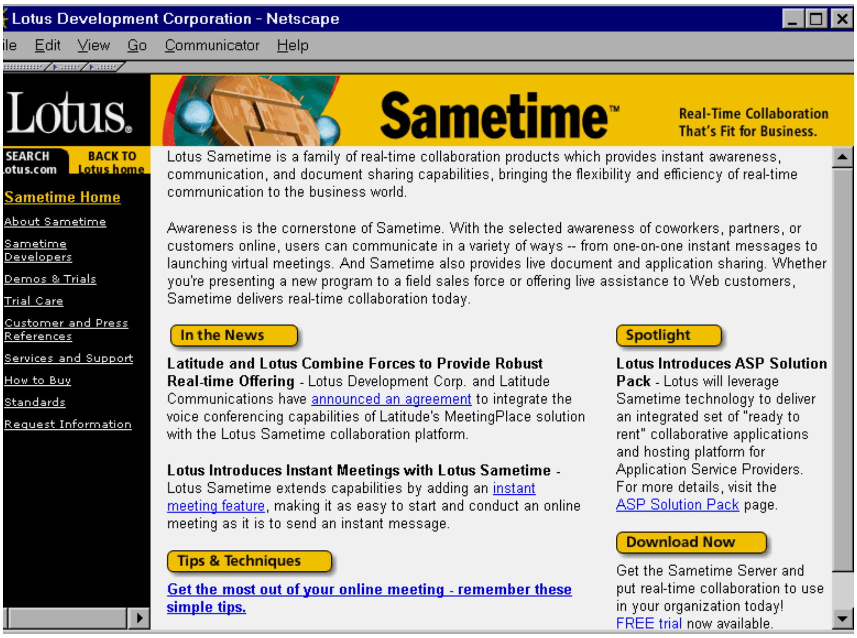
At this point, the service representative can take any of the actions indicated by the buttons at the center of the screen.

The service representative can send the customer a Web page. The available Web pages from which to choose are shown in the Url List drop-down box.



Here the two options are to send the customer a Databeam Web page or a Sametime Web page. When the page is sent to the customer, it is also sent to the service representative's screen to verify what page has been sent.

Here is how the screen looks if we send the customer the Sametime page:



Lotus Sametime is a family of real-time collaboration products which provides instant awareness, communication, and document sharing capabilities, bringing the flexibility and efficiency of real-time communication to the business world.

Awareness is the cornerstone of Sametime. With the selected awareness of coworkers, partners, or customers online, users can communicate in a variety of ways -- from one-on-one instant messages to launching virtual meetings. And Sametime also provides live document and application sharing. Whether you're presenting a new program to a field sales force or offering live assistance to Web customers, Sametime delivers real-time collaboration today.

In the News

Latitude and Lotus Combine Forces to Provide Robust Real-time Offering - Lotus Development Corp. and Latitude Communications have announced an agreement to integrate the voice conferencing capabilities of Latitude's MeetingPlace solution with the Lotus Sametime collaboration platform.

Lotus Introduces Instant Meetings with Lotus Sametime - Lotus Sametime extends capabilities by adding an instant meeting feature, making it as easy to start and conduct an online meeting as it is to send an instant message.

Tips & Techniques

Get the most out of your online meeting - remember these simple tips.

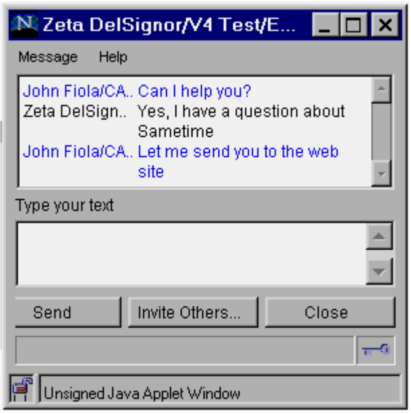
Spotlight

Lotus Introduces ASP Solution Pack - Lotus will leverage Sametime technology to deliver an integrated set of "ready to rent" collaborative applications and hosting platform for Application Service Providers. For more details, visit the ASP Solution Pack page.

Download Now

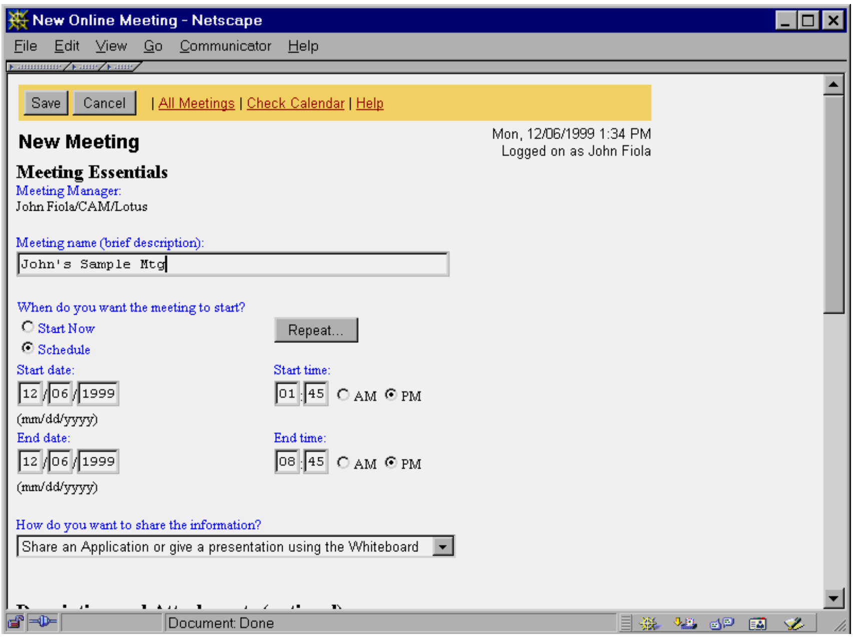
Get the Sametime Server and put real-time collaboration to use in your organization today! FREE trial now available.

The service representative can also send a message to the customer, by clicking the Instant Message button. Here the service representative can start a direct conversation with the customer.



John Fiola/CA.. Can I help you?
Zeta DelSign.. Yes, I have a question about Sametime
John Fiola/CA.. Let me send you to the web site

The service representative can also start a meeting with the customer. Prior to helping customers, the service representative clicks Create New Meeting at the bottom of the screen, and the service representative is sent to the Meeting Center, where they create a new meeting. The screen the service representative uses to create the meeting is shown in the next figure.


New Online Meeting - Netscape
File Edit View Go Communicator Help
Save Cancel | All Meetings | Check Calendar | Help

New Meeting

Mon, 12/06/1999 1:34 PM
Logged on as John Fiola

Meeting Essentials

Meeting Manager:
John Fiola/CAM/Lotus

Meeting name (brief description):
John's Sample Mtg

When do you want the meeting to start?
Start Now
[✓ Schedule]

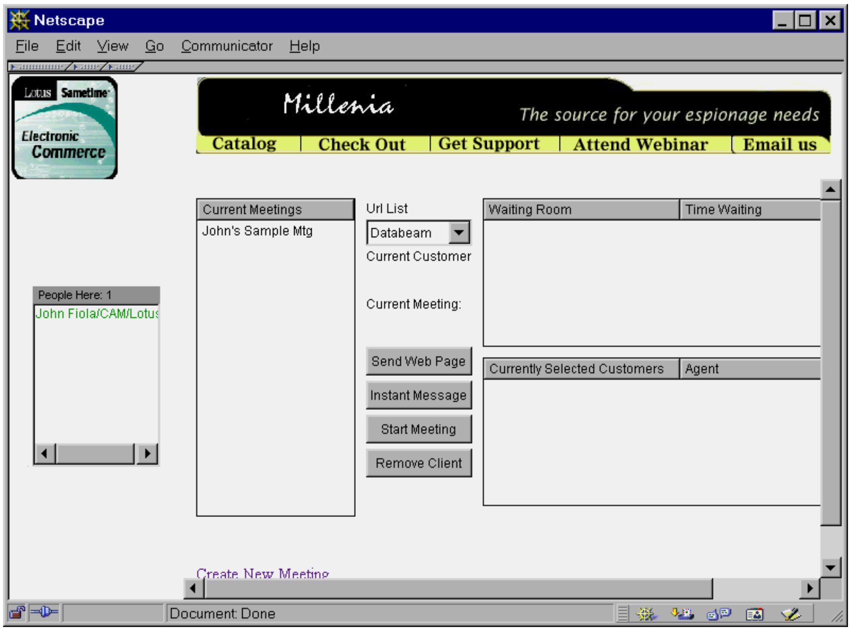
Repeat...

Start date: 12/06/1999 (mm/dd/yyyy)
Start time: 01:45 AM [✓ PM]
End date: 12/06/1999 (mm/dd/yyyy)
End time: 08:45 AM [✓ PM]

How do you want to share the information?
Share an Application or give a presentation using the Whiteboard

Document: Done

Once the meeting is created, it will be listed in the Current Meetings area as shown in the following figure.



People Here: 1
John Fiola/CAM/Lotus

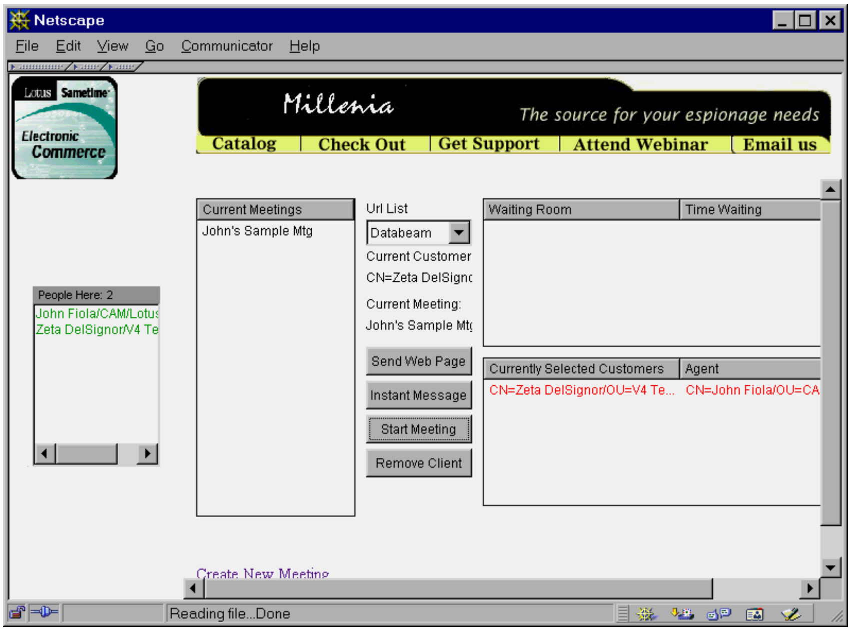
Current Meetings
John's Sample Mtg

Url List: Databeam
Current Customer
Current Meeting:

Waiting Room	Time Waiting

Currently Selected Customers	Agent

While the service representative is helping a customer, they can click the Start Meeting button.



Current Meetings
John's Sample Mtg

Url List: Databeam

Current Customer
CN=Zeta DelSignc

Current Meeting:
John's Sample Mtç

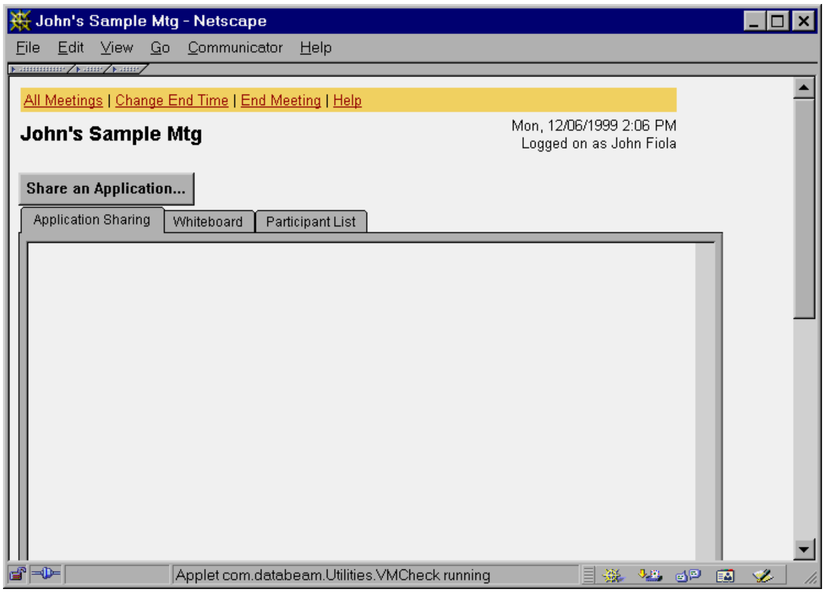
People Here: 2
John Fiola/CAM/Lotus
Zeta DelSignor/V4 Te

Waiting Room	Time Waiting

Currently Selected Customers	Agent
CN=Zeta DelSignor/OU=V4 Te...	CN=John Fiola/OU=CA

Reading file...Done

This causes the meeting setup feature to start on both the service representative's and the customer's machines. Now the service representative and the customer can share applications and draw on the whiteboard.

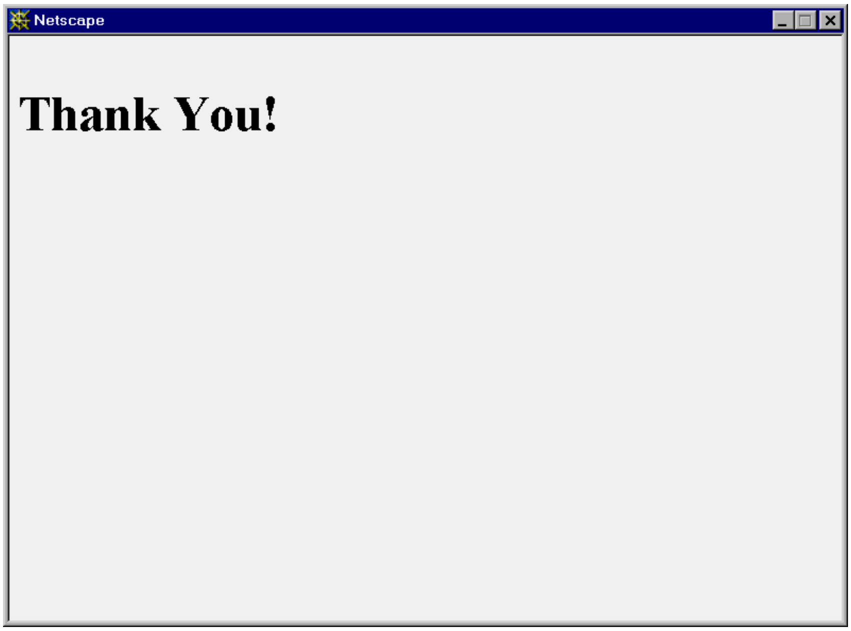

John's Sample Mtg - Netscape
File Edit View Go Communicator Help
All Meetings | Change End Time | End Meeting | Help
Share an Application...
Application Sharing
Whiteboard
Participant List

John's Sample Mtg

Mon, 12/06/1999 2:06 PM
Logged on as John Fiola

Applet com.databeam.Utilities.VMCheck running

Finally, when the Service representative is finished assisting the customer, they can end the session with this customer. The service representative clicks the Remove Client button. The customer is moved from the Currently Selected Customers list back to the Waiting Room list. A Thank You message is also sent to the customer's screen.



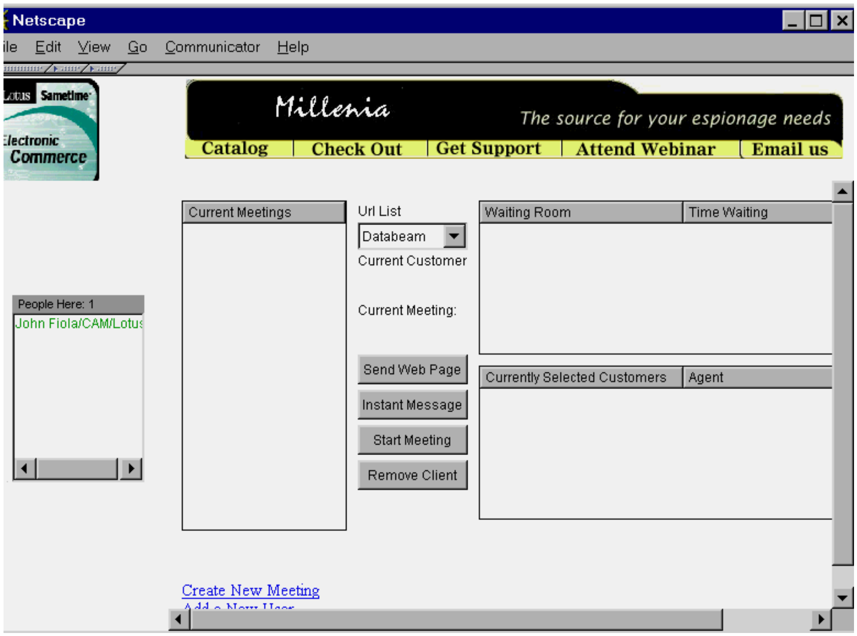
Thank You!

When the customer logs off from the site, they are removed from the Waiting Room and the service representative is ready to help the next person in line.

Now that we have looked at what the application does, we will show how it was created.

Customer service agent under the hood

We use two separate databases: an e-commerce site and a help center, both of which are available for download. In each database, we copy the necessary agents and scripts into the databases, as described in previous chapters. The customer service agent logs in to the e-commerce site through a browser. When they do, the Service Center Agent page is loaded into their browser. The page the agent sees is shown in the following figure.



Following is the source code for the page:

```
<FRAMESET ROWS="100,*" COLS="*" FRAMEBORDER="NO" BORDER="0"
FRAMESPACING="0">

 <FRAME SRC="/ecommerce/banner.html" SCROLLING="NO"
MARGINWIDTH="0" MARGINHEIGHT="0" NORESIZE FRAMEBORDER="NO">

 <FRAMESET COLS="168,*" ROWS="*" BORDER="0" FRAMESPACING="0"
FRAMEBORDER="NO">

 <FRAMESET COLS="*" ROWS="25%,75%" BORDER="0" FRAMESPACING="0"
FRAMEBORDER="NO">

  <FRAME src="/ecommerce.nsf/agentlogin?OpenForm" name="queue"
MARGINWIDTH=0 MARGINHEIGHT=0 NORESIZE FRAMEBORDER="NO"
SCROLLING="NO">

  <FRAME SRC="/ecommerce.nsf/toolbar2?openform" NAME="toolbar"
MARGINWIDTH=0 MARGINHEIGHT=0 NORESIZE FRAMEBORDER="NO"
SCROLLING="NO">

</FRAMESET>
```

```
 <FRAME SRC="/ecommerce.nsf/agentqueue2?OpenForm" NAME="main"
MARGINWIDTH=12 MARGINHEIGHT=0 SCROLLING="AUTO" NORESIZE
FRAMEBORDER="NO">

</FRAMESET>

</FRAMESET>

<NOFRAMES>

<BODY>

</BODY></NOFRAMES>
```

With this code, we can load multiple frames into the frameset. The two frames that contain the user interface described in the previous section are the following:

PeopleHere Applet
This is the list on the left side of the screen which shows all the people logged in to the site. The HTML call for this is:

```
<FRAME SRC="/ecommerce.nsf/toolbar2?openform...
```

Customer Service Applet
This is the control center for the customer service representatives. The HTML for this is:

```
<FRAME SRC="/ecommerce.nsf/agentQueue2?OpenForm...
```

These agents load two different forms stored in our e-commerce database. The Agents and Script libraries that we have discussed in previous chapters have already been added to the database. We examine each of these forms.

People Here applet

This applet is loaded with the HTML call:

```
    <FRAME SRC="/ecommerce.nsf/toolbar2?openform...
```

This loads a form from the database called "toolbar2". We look at the features of this form that are added to run this applet.

First, we add the formula to run agents to the WebQueryOpen event.

```
    @Command([ToolsRunMacro]; "SametimeWebQueryOpen");

    @Command([ToolsRunMacro]; "SametimeGetHost")
```

Next, we set up the necessary fields and agents in the form as follows:

- USER_NAME
- HTTP_User_Agent

Next, we enter the following HTML code for the Customer Service applet on the form:

```
<applet name='vpApplet' code='STCommunityApplet.class'
codebase='/ecommerce/applet' mayscript width=120 height=170
VIEWASTEXT>
<PARAM NAME='archive'  VALUE='/stapi/VpAPI.jar'>
<PARAM NAME='cabbase'  VALUE='/stapi/VpAPI.cab'>
<param name="cabbase" value="VpApi.cab, ecommunity.jar">
<param name='displayWIHList' value='true'>
<param name='counterFont' value='Arial,plain,11'>
<param name='counterBGColor' value='153,153,153'>
<param name='counterFGColor' value='black'>
<param name='counterTitle' value='People Here'>
<param name='peopleListFont' value='Arial,plain,12'>
<param name='peopleListBGColor' value='250,250,250'>
<param name='peopleListFGColor' value='0,153,0'>
<param name='noAwarenessBGColor' value='243,252,231'>
<param name='loginName' value='[LOGIN_NAME]'>
<param name='token' value='[SAMETIME_TOKEN]'>
<param name="tunnelport" value="8082">
</applet>
```

Here, we add some of the fields so that the values become parameters for the applet. In this instance, we have two fields:

- LOGIN_NAME
- SAMETIME_TOKEN

The LOGIN_NAME field refers to the User_name field set at the top of the form. The SAMETIME_TOKEN field contains the token to log in to the Sametime server. This is set by the SametimeWebQueryOpen agent.

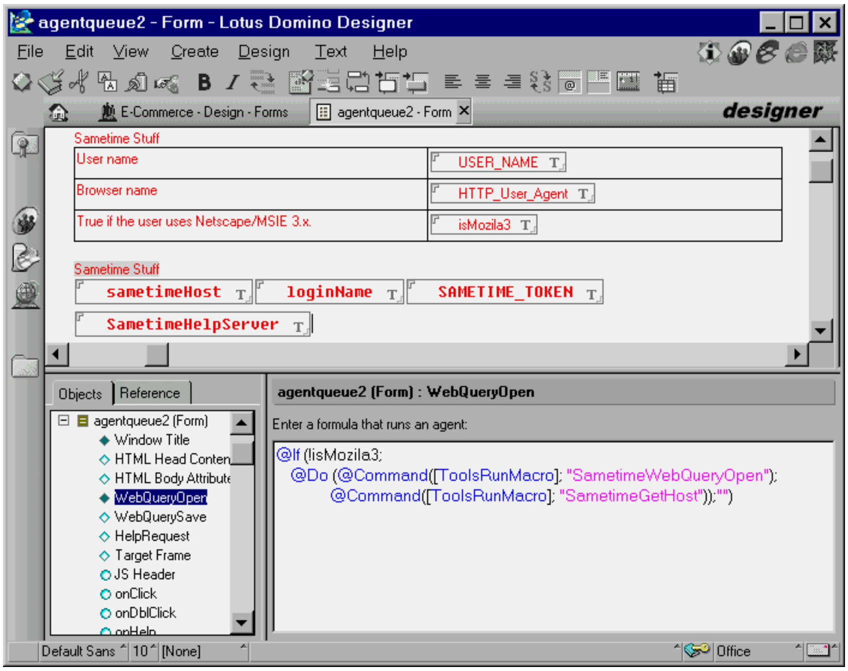
Customer Service applet

This applet is loaded with the HTML call:

```
<FRAME SRC="/ecommerce.nsf/agentqueue2?openform...
```

This loads a form from the database called "agentqueue2". We look at the features of this form that are added to run this applet. The source code for this applet is available for download at `http://www.redbooks.ibm.com`

We set up the necessary fields and agents in the form as shown in the figure below.



Sametime Stuff	
User name	USER_NAME T
Browser name	HTTP_User_Agent T
True if the user uses Netscape/MSIE 3.x.	isMozila3 T

Sametime Stuff

sametimeHost T loginName T SAMETIME_TOKEN T

SametimeHelpServer T

agentqueue2 (Form) : WebQueryOpen

Enter a formula that runs an agent:

```
@If (!isMozila3;
  @Do (@Command([ToolsRunMacro]; "SametimeWebQueryOpen");
       @Command([ToolsRunMacro]; "SametimeGetHost"));"")
```

Next, we place the HTML code for the Customer Service applet on the form:

```
<center>
<applet name = “AgentView” code = “AgentView.class”
codebase = “http://<Computed Value>/HelpCenter”
     width=700 height=325>
<param name = ’archive’  VALUE=’/stapi/VpAPI.jar’>
<param name = ’cabbase’  VALUE=’/stapi/VpAPI.cab’>
<param name = “Username” value=“<Computed Value>”>
<param name = “Token”   value=“<Computed Value>”>
<param name = “Server”  value=“<Computed Value>”>
<param name = “Place”   value=“HelpCenter”>
<param name = “StartMeetingButton”   value = “Start Meeting”>
<param name = “InstantMessageButton”value = “Instant Message”>
<param name = “SendUrlButton” value = “Send Web Page”>
<param name = “RemoveClientButton” value = “Remove Client”>
```

```
<param name = “MeetingListLabel” value = “Current Meetings”>
<param name = “SelectedListLabel” value = “Currently Selected
Customers”>
<param name = “WaitingListLabel” value = “Waiting Room”>
<param name = “UrlListLabel” value = “Url List”>
<param name = “MeetingListRefresh” value = “5”>
<param name = “CurrentCustomerLabel” value = “Current
Customer:”>
<param name = “CurrentMeetingLabel” value = “Current Meeting:”>
<param name = “GoodByeURL” value=“localhost”>
<param name = “TunnelPort” value = “8082”>
<param name = “EmptyURL” value = “http://<Computed
Value>/helpcenter.nsf/blank?openform”>
<param name = “Url0” value=“Databeam;http://www.databeam.com”>
<param name = “Url1”
value=“Sametime;http://www.lotus.com/sametime”>
</applet>
</center>
<br clear=“all”>
<a href=“http://<Computed
Value>/STConf.nsf/frmConference?Openform”
target=“newmeeting”>Create New Meeting</a><br>
<a href=“http://<Computed Value>/Streg.nsf”
target=“newuser”>Add a New User</a><br>
<a href=“http://<Computed Value>/helpcenter.nsf/help?openform”
target = “help”>Help</a><br>
</td></tr></table>
</td></tr></table>
</center>
```

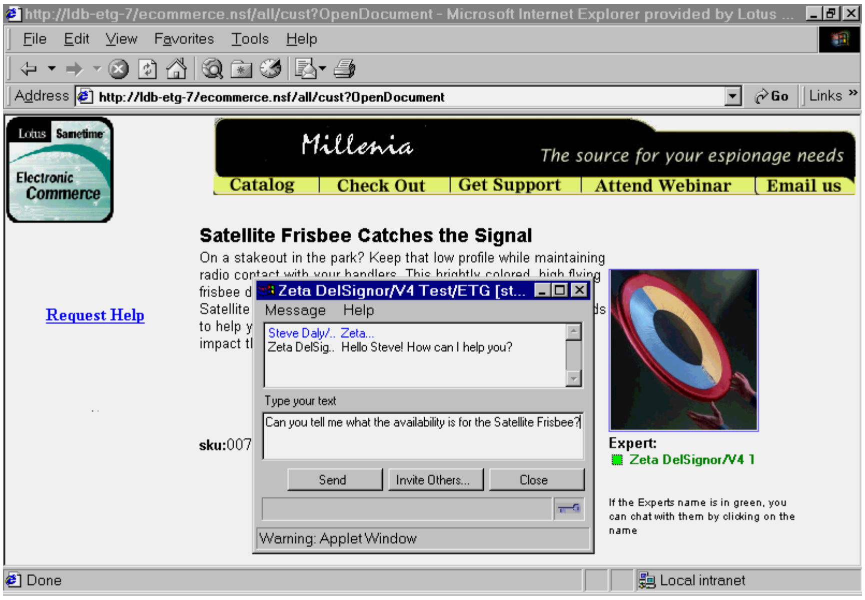
For each of the <Computed Value> entries in the applet parameters, we add computed text and reference the Sametime fields on the form. For example, for the Token parameter, the Computed text value is the “SAMETIME_TOKEN” field.

We also hard-code the place parameter of the applet with the value HelpCenter. This allows us to have a place which is independent of the database and the document, and allows us to have a separate place where customers can actively request help.

The customer side

Millenia wanted to provide customers with the ability to browse its product catalog via the Web. One of Millenia's goals is to provide unparalleled customer service, so Millenia decided to integrate Sametime into their Web site. Customers can now request help from customer service agents or determine when a product expert is online to answer questions.

The next figure shows the Sametime-enabled product catalog. The name under the product picture shows the customer that a product expert is online. The customer can then initiate a chat with the expert to ask questions.


http://ldb-etg-7/ecommerce.nsf/all/cust?OpenDocument - Microsoft Internet Explorer provided by Lotus ...
File Edit View Favorites Tools Help
Address http://ldb-etg-7/ecommerce.nsf/all/cust?OpenDocument
Go Links
Lotus Sametime Electronic Commerce
Millenia
The source for your espionage needs
Catalog Check Out Get Support Attend Webinar Email us
Request Help
Zeta DelSignor/V4 Test/ETG [st...
Message Help
Type your text
Send Invite Others... Close
Warning: Applet Window
Done
Local intranet

Satellite Frisbee Catches the Signal

On a stakeout in the park? Keep that low profile while maintaining radio contact with your handlers. This brightly colored, high flying frisbee d

Satellite

to help y

impact tl

sku:007

Steve Daly/.. Zeta...
Zeta DelSig.. Hello Steve! How can I help you?

Can you tell me what the availability is for the Satellite Frisbee?

Expert:
🟩 Zeta DelSignor/V4 1

If the Experts name is in green, you can chat with them by clicking on the name

The catalog also gives the customer access to the Millenia Help Center. By clicking the Request Help link, customers are queued up in the Waiting Room area of the customer service agent application that was discussed earlier in this chapter.

Sametime-enabling the catalog item form

We created the catalog item form using Domino Designer. The USER_NAME and SAMETIME_TOKEN fields were added to the form's design. We also copied the script libraries and agents to their appropriate locations. The QueryOpen and WebQueryOpen events were also updated to get the appropriate authentication token that would allow us to log in to the Sametime server.

The LiveNames applet

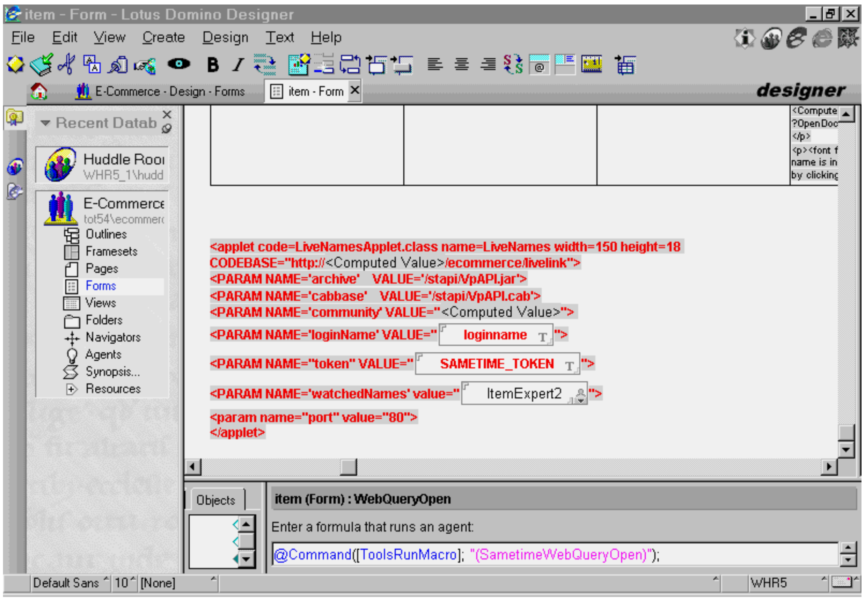
Displaying the product expert is done with a Java applet built with the Community Services API. The LiveNames applet is available on the Sametime server as an example of developing with the toolkit. It is an excellent example of using the Who Is Online service.

When a customer opens up a product page, the LiveNames applet registers the customer in Millenia's Sametime Community. The applet then checks to see if the product expert is online by passing the name as a parameter. If the expert is online, the name turns green. When the customer has a question for the expert, he double-clicks the name and a chat is initiated.

Adding the LiveNames applet

To include the applet, we added an <APPLET> tag to the catalog item form. We placed the Java files in a subdirectory on the Domino R5 server.

Note We could have imported the Java files into the application as a Shared Resource but we decided to use <APPLET> tags to show another method of adding the LiveNames applet to the form. This also allowed us to revise the applet during development without having to refresh the Domino application.



```
<applet code=LiveNamesApplet.class name=LiveNames width=150 height=18
CODEBASE="http://<Computed Value>/ecommerce/livelink">
<PARAM NAME='archive'   VALUE='/stapi/VpAPI.jar'>
<PARAM NAME='cabbase'   VALUE='/stapi/VpAPI.cab'>
<PARAM NAME='community' VALUE="<Computed Value>">
<PARAM NAME='loginName' VALUE=" loginname ">
<PARAM NAME="token" VALUE=" SAMETIME_TOKEN ">
<PARAM NAME='watchedNames' value=" ItemExpert2 ">
<param name="port" value="80">
</applet>
```

item (Form) : WebQueryOpen

Enter a formula that runs an agent:

```
@Command([ToolsRunMacro]; "(SametimeWebQueryOpen)");
```

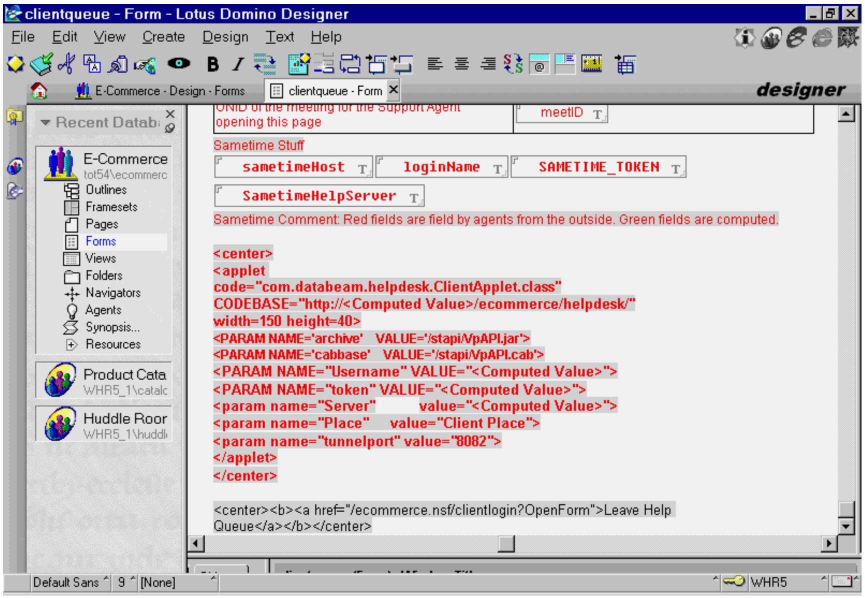
The LiveNames applet uses several parameters for operation:

Parameter Name	*Definition*
archive	String. Name of the .JAR archive that contains the Community Services API.
cabbase	String. Name of the .CAB archive that contains the Community Services API.
community	String. Name of the Sametime server.
loginName	String. Username to log in to the community.
SAMETIME_TOKEN	String. Authentication token.
watchedNames	String. List of names to check for online status.
port	String. TCP/IP port number used to connect to Sametime server.

The Help Status agent

There is one other Java applet that we put on the catalog site. We built an applet that communicates the customer's status in the support queue once they click Request Help link.

The applet uses the Who Is Here Service of the Sametime API and checks the Client Place place in the Sametime Community. The following figure shows the <APPLET> tag that was put onto the form used in the site.



opening this page
meetID

Sametime Stuff

sametimeHost loginName SAMETIME_TOKEN

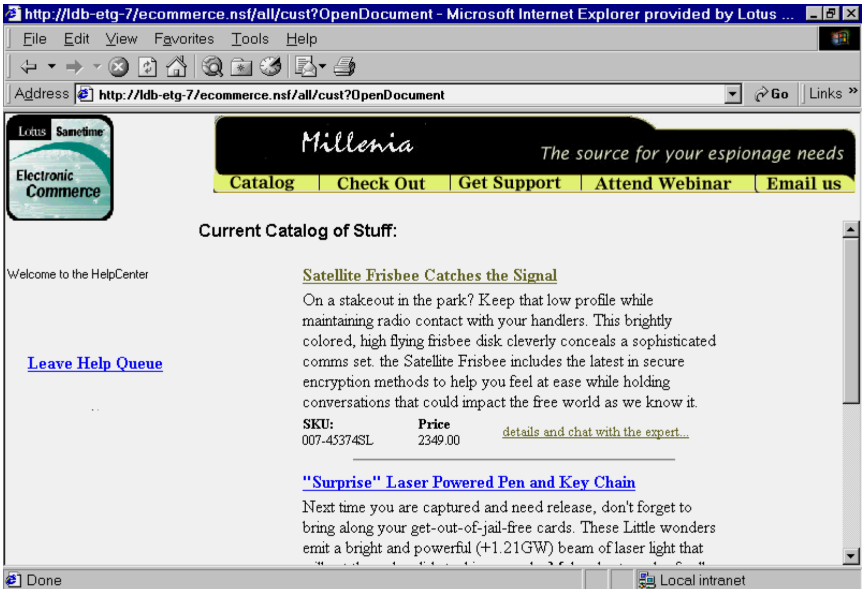
SametimeHelpServer

Sametime Comment: Red fields are field by agents from the outside. Green fields are computed.

```
<center>
<applet
code="com.databeam.helpdesk.ClientApplet.class"
CODEBASE="http://<Computed Value>/ecommerce/helpdesk/"
width=150 height=40>
<PARAM NAME='archive'   VALUE='/stapi/VpAPI.jar'>
<PARAM NAME='cabbase'   VALUE='/stapi/VpAPI.cab'>
<PARAM NAME="Username" VALUE="<Computed Value>">
<PARAM NAME="token" VALUE="<Computed Value>">
<param name="Server"        value="<Computed Value>">
<param name="Place"     value="Client Place">
<param name="tunnelport" value="8082">
</applet>
</center>

<center><b><a href="/ecommerce.nsf/clientlogin?OpenForm">Leave Help
Queue</a></b></center>
```

This applet logs the user in to the Sametime Community and enters the Client Place place. Customer service agents enter this place when they enter the agent applet that was discussed earlier in this chapter. The place is used to communicate between the agent and the customer. Status messages are sent as the agent accepts a customer from the help queue.

The following figures demonstrate how the Help Status agent works. Once the customer clicks the Request Help link, the applet starts and welcomes the customer to the Help Center. This tells the customer that he has been added to the customer service queue.


http://ldb-etg-7/ecommerce.nsf/all/cust?OpenDocument - Microsoft Internet Explorer provided by Lotus ...
File Edit View Favorites Tools Help
Address http://ldb-etg-7/ecommerce.nsf/all/cust?OpenDocument
Go Links
Lotus Sametime Electronic Commerce
Millenia
The source for your espionage needs
Catalog | Check Out | Get Support | Attend Webinar | Email us
Done
Local intranet

Current Catalog of Stuff:

Welcome to the HelpCenter

Leave Help Queue

Satellite Frisbee Catches the Signal

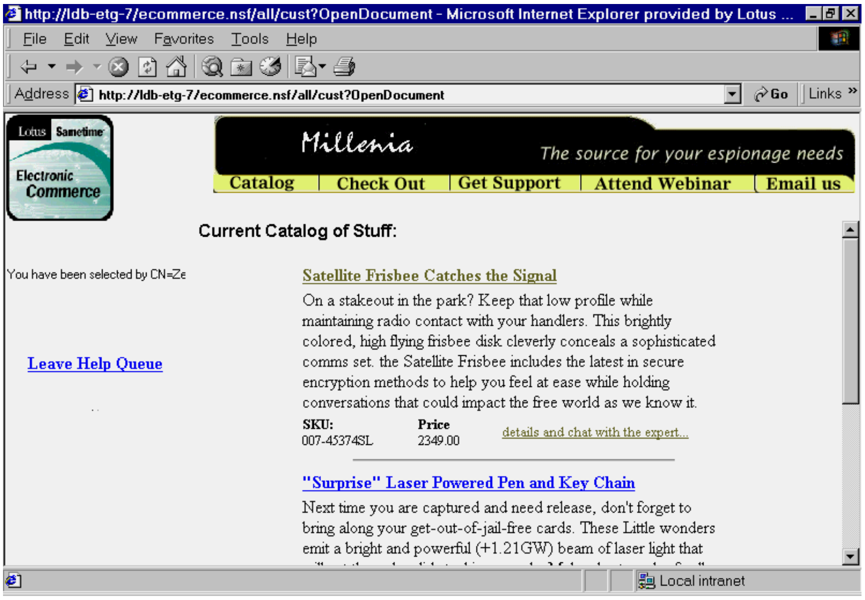
On a stakeout in the park? Keep that low profile while maintaining radio contact with your handlers. This brightly colored, high flying frisbee disk cleverly conceals a sophisticated comms set. the Satellite Frisbee includes the latest in secure encryption methods to help you feel at ease while holding conversations that could impact the free world as we know it.

SKU:	Price	
007-45374SL	2349.00	details and chat with the expert...

"Surprise" Laser Powered Pen and Key Chain

Next time you are captured and need release, don't forget to bring along your get-out-of-jail-free cards. These Little wonders emit a bright and powerful (+1.21GW) beam of laser light that

When the customer is selected from the queue by a customer service agent, the applet responds:


http://ldb-etg-7/ecommerce.nsf/all/cust?OpenDocument - Microsoft Internet Explorer provided by Lotus ...
File Edit View Favorites Tools Help
Address http://ldb-etg-7/ecommerce.nsf/all/cust?OpenDocument
Go Links
Lotus Sametime
Electronic Commerce
Millenia
The source for your espionage needs
Catalog
Check Out
Get Support
Attend Webinar
Email us
Local intranet

Current Catalog of Stuff:

You have been selected by CN=Ze

Leave Help Queue

Satellite Frisbee Catches the Signal

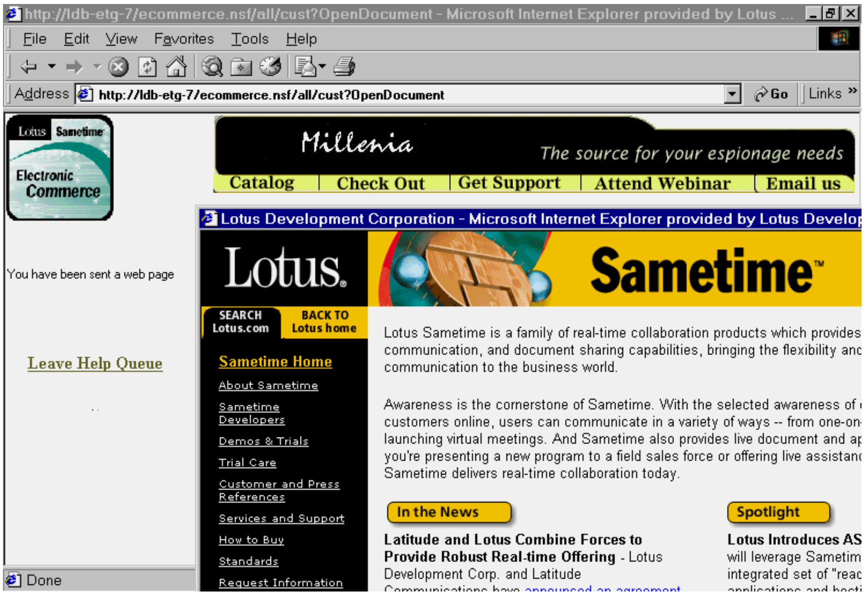
On a stakeout in the park? Keep that low profile while maintaining radio contact with your handlers. This brightly colored, high flying frisbee disk cleverly conceals a sophisticated comms set. the Satellite Frisbee includes the latest in secure encryption methods to help you feel at ease while holding conversations that could impact the free world as we know it.

SKU:	Price	
007-45374SL	2349.00	details and chat with the expert...

"Surprise" Laser Powered Pen and Key Chain

Next time you are captured and need release, don't forget to bring along your get-out-of-jail-free cards. These Little wonders emit a bright and powerful (+1.21GW) beam of laser light that

The Help Status agent then reports the activities between the customer service agent and the customer (for example, when the agent sends the customer a Web page to view).


http://ldb-etg-7/ecommerce.nsf/all/cust?OpenDocument - Microsoft Internet Explorer provided by Lotus ...
File Edit View Favorites Tools Help
Address http://ldb-etg-7/ecommerce.nsf/all/cust?OpenDocument
Go Links
Lotus Sametime
Electronic Commerce
Millenia
The source for your espionage needs
Catalog
Check Out
Get Support
Attend Webinar
Email us
Done
Lotus Development Corporation - Microsoft Internet Explorer provided by Lotus Develop
Lotus.
Sametime™
SEARCH Lotus.com
BACK TO Lotus home
Sametime Home
About Sametime
Sametime Developers
Demos & Trials
Trial Care
Customer and Press References
Services and Support
How to Buy
Standards
Request Information
In the News
Spotlight

You have been sent a web page

Leave Help Queue

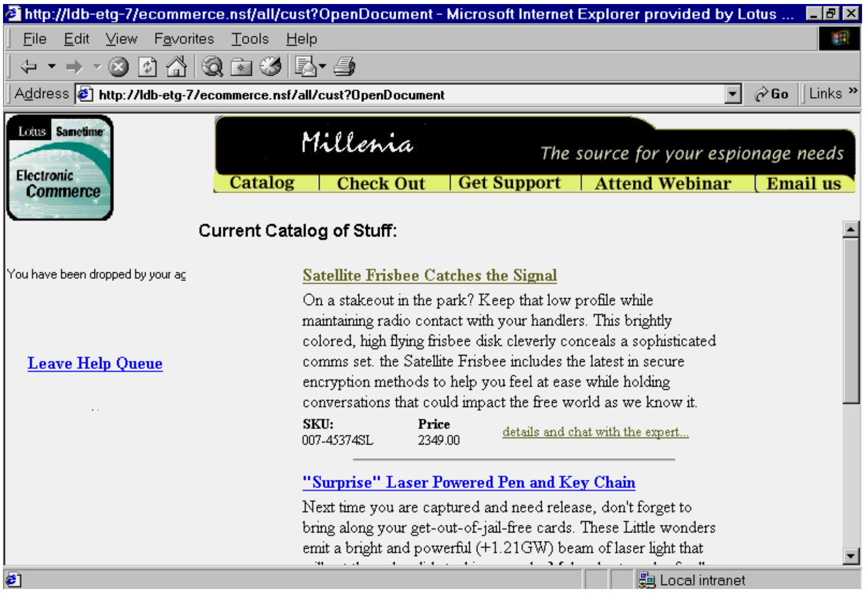
Lotus Sametime is a family of real-time collaboration products which provides communication, and document sharing capabilities, bringing the flexibility and communication to the business world.

Awareness is the cornerstone of Sametime. With the selected awareness of customers online, users can communicate in a variety of ways -- from one-on- launching virtual meetings. And Sametime also provides live document and ap you're presenting a new program to a field sales force or offering live assistanc Sametime delivers real-time collaboration today.

Latitude and Lotus Combine Forces to Provide Robust Real-time Offering - Lotus Development Corp. and Latitude

Lotus Introduces AS will leverage Sametim integrated set of "reac

And finally, the Help Status applet reports when the customer service agent has finished and removed the customer from their queue.


http://ldb-etg-7/ecommerce.nsf/all/cust?OpenDocument - Microsoft Internet Explorer provided by Lotus ...
File Edit View Favorites Tools Help
Address http://ldb-etg-7/ecommerce.nsf/all/cust?OpenDocument
Go
Links
Lotus Sametime
Electronic Commerce
Millenia
The source for your espionage needs
Catalog
Check Out
Get Support
Attend Webinar
Email us
Local intranet

Current Catalog of Stuff:

You have been dropped by your ag

Leave Help Queue

Satellite Frisbee Catches the Signal

On a stakeout in the park? Keep that low profile while maintaining radio contact with your handlers. This brightly colored, high flying frisbee disk cleverly conceals a sophisticated comms set. the Satellite Frisbee includes the latest in secure encryption methods to help you feel at ease while holding conversations that could impact the free world as we know it.

SKU:	Price	
007-45374SL	2349.00	details and chat with the expert...

"Surprise" Laser Powered Pen and Key Chain

Next time you are captured and need release, don't forget to bring along your get-out-of-jail-free cards. These Little wonders emit a bright and powerful (+1.21GW) beam of laser light that

Special notices

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these ITSO publications, see "How To Get IBM Redbooks."

- *Lotus Domino Release 5.0: A Developer's Handbook,* IBM form number SG24-5331-01, Lotus part number CC7EDNA
- *Connecting Domino to the Enterprise Using Java*, IBM form number SG24-5425, Lotus part number CT6EMNA
- *LotusScript for Visual Basic Programmers*, IBM form number SG24-4856, Lotus part number 12498
- *Developing Web Applications Using Lotus Notes Designer for Domino 4.6*, IBM form number SG24-2183, Lotus part number 12974
- *Lotus Notes 4.5: A Developers Handbook*, IBM form number SG24-4876, Lotus part number AA0425
- *Lotus Solutions for the Enterprise, Volume 1. Lotus Notes: An Enterprise Application Platform*, IBM form number SG24-4837, Lotus part number 12968
- *Lotus Solutions for the Enterprise, Volume 2. Using DB2 in a Domino Environment*, IBM form number SG24-4918, Lotus part number CT69BNA
- *Lotus Solutions for the Enterprise, Volume 3. Using the IBM CICS Gateway for Lotus Notes*, IBM form number SG24-4512
- *Lotus Solutions for the Enterprise, Volume 4. Lotus Notes and the MQSeries Enterprise Integrator*, IBM form number SG24-2217, Lotus part number 12992
- *Lotus Solutions for the Enterprise, Volume 5. NotesPump, the Enterprise Data Mover*, IBM form number SG24-5255, Lotus part number CT69DNA
- *Enterprise Integration with Domino for S/390,* IBM form number SG24-5150

Other Lotus-related IBM publications

The publications listed in this section may also be of interest:

- *A Roadmap for Deploying Domino in the Organization*, IBM form number SG24-5617, Lotus part number CT6P8NA
- *The Three Steps to Super.Human.Software: Compare, Coexist, Migrate; From Microsoft Exchange to Lotus Domino, Part One: Comparison*, IBM form number SG24-5614, Lotus part number CT7QTNA
- *The Three Steps to Super.Human.Software: Compare, Coexist, Migrate; From Microsoft Exchange to Lotus Domino, Part Two: Coexistence and Migration*, IBM form number SG24-5615, Lotus part number CT7QWNA
- *Lotus Notes and Domino R5.0 Security Infrastructure Revealed,* IBM form number SG24-5341, Lotus part number CT6TPNA
- *Lotus Notes and Domino: The Next Generation in Messaging. Moving from Microsoft Mail to Lotus Notes and Domino*, IBM form number SG24-5152, Lotus part number CT7SBNA
- *Eight Steps to a Successful Messaging Migration: A Planning Guide for Migrating to Lotus Notes and Domino*, IBM form number SG24-5335, Lotus part number CT6HINA
- *Deploying Domino in an S/390 Environment*, IBM form number SG24-2182, Lotus part number 12957
- *The Next Step in Messaging: Case Studies on Lotus cc:Mail to Lotus Domino and Lotus Notes*, IBM form number SG24-5100, Lotus part number 12992
- *Lotus Notes and Domino: The Next Generation in Messaging. Moving from Novell GroupWise to Lotus Notes and Domino*, IBM form number SG24-5321, Lotus part number CT7NNNA
- *High Availability and Scalability with Domino Clustering and Partitioning on Windows NT*, IBM form number SG24-5141, Lotus part number CT6XMIE
- *From Client/Server to Network Computing, A Migration to Domino*, IBM form number SG24-5087, Lotus part number CT699NA
- *Netfinity and Domino R5.0 Integration Guide*, IBM form number SG24-5313, Lotus part number CT7BKNA
- *Lotus Domino R5 for IBM RS/6000*, IBM form number SG24-5138, Lotus part number CT7BHNA
- *Lotus Domino Release 4.6 on IBM RS/6000: Installation, Customization and Administration*, IBM form number SG24-4694, Lotus part number 12969
- *High Availability and Scalability with Domino Clustering and Partitioning on AIX*, IBM form number SG24-5163, Lotus part number CT7J0NA
- *Lotus Domino for AS/400: Installation, Customization and Administration*, IBM form number SG24-5181, Lotus part number AA0964

- *Lotus Domino for S/390 Release 4.6: Installation, Customization & Administration*, IBM form number SG24-2083
- *Lotus Domino for S/390 Performance Tuning and Capacity Planning*, IBM form number SG24-5149, Lotus part number CT6XNIE
- *Porting C Applications to Lotus Domino on S/390*, IBM form number SG24-2092, Lotus part number AB1720
- *Managing Domino/Notes with Tivoli Manager for Domino, Enterprise Edition, Version 1.5*, IBM form number SG24-2104
- *Measuring Lotus Notes Response Times with Tivoli's ARM Agents*, IBM form number SG24-4787, Lotus part number CT6UKIE
- *Using ADSM to Back Up Lotus Notes*, IBM form number SG24-4534
- *Performance Considerations for Domino Applications*, IBM form number SG24-5602, Lotus part number CT7V6NA.
- *Lotus Domino Release 5.0: A Developer's Handbook*, IBM form number SG24-5331-01, Lotus part number CC7EDNA.

IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at **`http://www.redbooks.ibm.com/`** for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	*Collection Kit Number*
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
Application Development Redbooks Collection	SK2T-8037
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr Format)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
AS/400 Redbooks Collection	SK2T-2849
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Networking and Systems Management Redbooks Collection	SK2T-6022
System/390 Redbooks Collection	SK2T-2177
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** **`http://www.redbooks.ibm.com`**

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the IBM Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are redbooks in progress; not all redpieces become redbooks and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States:	usib6fpl@ibmmail.com
Outside North America:	Contact information is in the "How to Order" section at this site: **`http://www.elink.ibmlink.ibm.com/pbl/pbl/`**

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the "How to Order" section at this site: **`http://www.elink.ibmlink.ibm.com/pbl/pbl/`**

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada (toll free)	1-800-267-4455
Outside North America	Fax phone number is in the "How to Order" section at this site: **`http://www.elink.ibmlink.ibm.com/pbl/pbl/`**

The latest information for customers may be found at the IBM Redbooks Web site.

IBM intranet for employees

IBM employees may register for information on workshops, residencies, and redbooks by accessing the IBM intranet Web site at **`http://w3.itso.ibm.com/`** and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access **`MyNews`** at **`http://w3.ibm.com/`** for redbook, residency, and workshop announcements.

IBM Redbook fax order form

Please send me the following:

Title	**Order Number**	**Quantity**

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

❏ Invoice to customer number

❏ Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, MasterCard, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Index

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook “made the difference” in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate**.

- Use the online **Contact us** review redbook form found at **`http://www.redbooks.ibm.com/`**
- Fax this form to: USA International Access Code +1 914 432 8264
- Send your comments in an Internet note to **`redbook@us.ibm.com`**

Document Number **Redbook Title**	SG24-5651-00 Lotus Sametime Application Development Guide
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction?	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	 O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM’s privacy policy?	The following link explains how we protect your personal information. **`http://www.ibm.com/privacy/yourprivacy/`**

Printed in the U.S.A.

SG24-5651-00

Part No. CT7AKNA

Lotus Sametime Application Development Guide

SG24-5651-00